



**Faculty of Engineering and Technology**  
**Electrical and Computer Engineering Department**

**ENCS3320-Computer Networks**

**Project#1**

Prepared by:

<b>Aws Shaheen</b>	<b>1212585</b>	<b>sec. 2</b>
<b>Alhasan Manasrah</b>	<b>1211705</b>	<b>sec. 2</b>
<b>Ghassan Qandeel</b>	<b>1212397</b>	<b>sec. 1</b>

Instructors: Mohammad Jubran & Abdalkarim Awad

Deadline: 22-12-2023

## Table of Contents

<b>Part 1:</b>	<b>3</b>
1) <i>In your own words, what are ping, tracert, nslookup, and telnet:</i>	<b>3</b>
2) <i>Make sure that your computer is connected to the internet and then run the following commands: ...</i>	<b>3</b>
a) <i>Ping a device in the same network, e.g. from a laptop to a smartphone:</i>	3
b) <i>ping www.cornell.edu:</i>	5
c) <i>tracert www.cornell.edu:</i>	6
d) <i>nslookup www.cornell.edu:</i>	7
3) <i>use wireshark to capture some DNS messages:</i>	<b>8</b>
<b>Part2: socket programing using python:</b>	<b>10</b>
<b>part3:</b>	<b>13</b>
0) <i>from rfce2616, what is Content-Type in the HTTP request and why do we need it?</i>	<b>13</b>
1) <i>if the request is / or /index.html or /main_en.html or /en (for example localhost:9966/ or localhost:9966/en) then the server should send main_en.html file with Content-Type: text/html:</i>	<b>14</b>
2) <i>If the request is /ar then the server response with main_ar.html which is an Arabic version of main_en.html:</i>	<b>16</b>
3) <i>if the request is an .html file then the server should send the requested html file with Content-Type: text/html. You can use any html file:</i>	<b>18</b>
4) <i>if the request is a .css file then the server should send the requested css file with Content-Type: text/css. You can use any CSS file:</i>	<b>19</b>
5) <i>if the request is a .png then the server should send the png image with Content-Type: image/png. You can use any image</i>	<b>20</b>
6) <i>if the request is a .jpg then the server should send the jpg image with ContentType: image/jpeg. You can use any image:</i>	<b>21</b>
7) <i>Use the status code 307 Temporary Redirect to redirect the following</i>	<b>22</b>
a) <i>If the request is /cr then redirect to cornell.edu website:</i>	22
b) <i>If the request is /so then redirect to stackoverflow.com website:</i>	23
c) <i>If the request is /rt then redirect to ritaj website:</i>	24
8) <i>If the request is wrong or the file doesn't exist the server should return a simple HTML webpage that contains (Content-Type: text/html):</i>	<b>25</b>
<b>Python code</b>	<b>26</b>
<b>Html code</b>	<b>30</b>
<b>Css code</b>	<b>32</b>

## Part 1:

1) *In your own words, what are ping, tracert, nslookup, and telnet:*

**Ping:** is a computer network administration software utility used to test the reachability of a host on an Internet Protocol network.

**Tracert:** computer network diagnostic commands for displaying possible routes and measuring transit delays of packets across an Internet Protocol network.

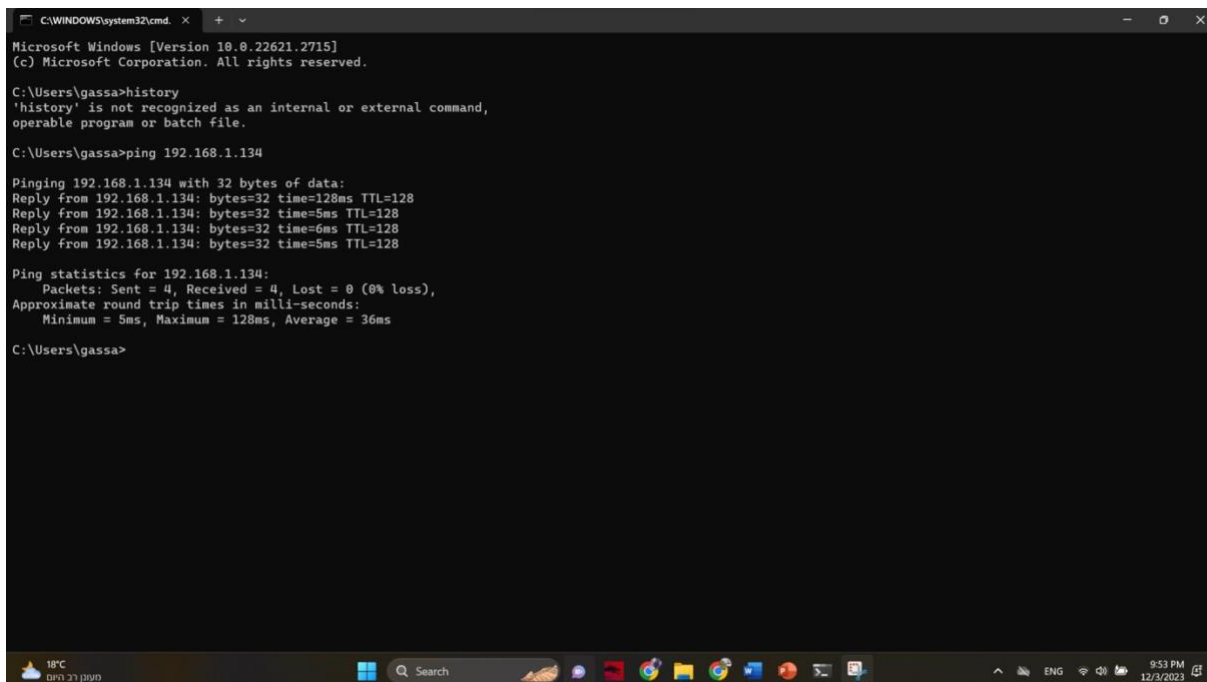
**Nslookup:** is a network administration command-line tool for querying the Domain Name System to obtain the mapping between domain name and IP address, or other DNS records.

**telnet:** is a client/server application protocol that provides access to virtual terminals of remote systems on local area networks or the Internet.

2) *Make sure that your computer is connected to the internet and then run the following commands:*

a) *Ping a device in the same network, e.g. from a laptop to a smartphone:*

Here is the result when we requested from first device to second device by writing ping 192.2.134 on the cmd line window.



```
C:\WINDOWS\system32\cmd. X + -
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\gassa>history
'history' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\gassa>ping 192.168.1.134

Pinging 192.168.1.134 with 32 bytes of data:
Reply from 192.168.1.134: bytes=32 time=128ms TTL=128
Reply from 192.168.1.134: bytes=32 time=5ms TTL=128
Reply from 192.168.1.134: bytes=32 time=6ms TTL=128
Reply from 192.168.1.134: bytes=32 time=5ms TTL=128

Ping statistics for 192.168.1.134:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 5ms, Maximum = 128ms, Average = 36ms

C:\Users\gassa>
```

Here is the result responding from the second device by writing ipconfig command on cmd line window.

```
C:\Windows\system32\cmd.exe
C:\Users\RUZAN>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Wireless LAN adapter Ethernet* 11:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Wireless LAN adapter Ethernet* 12:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

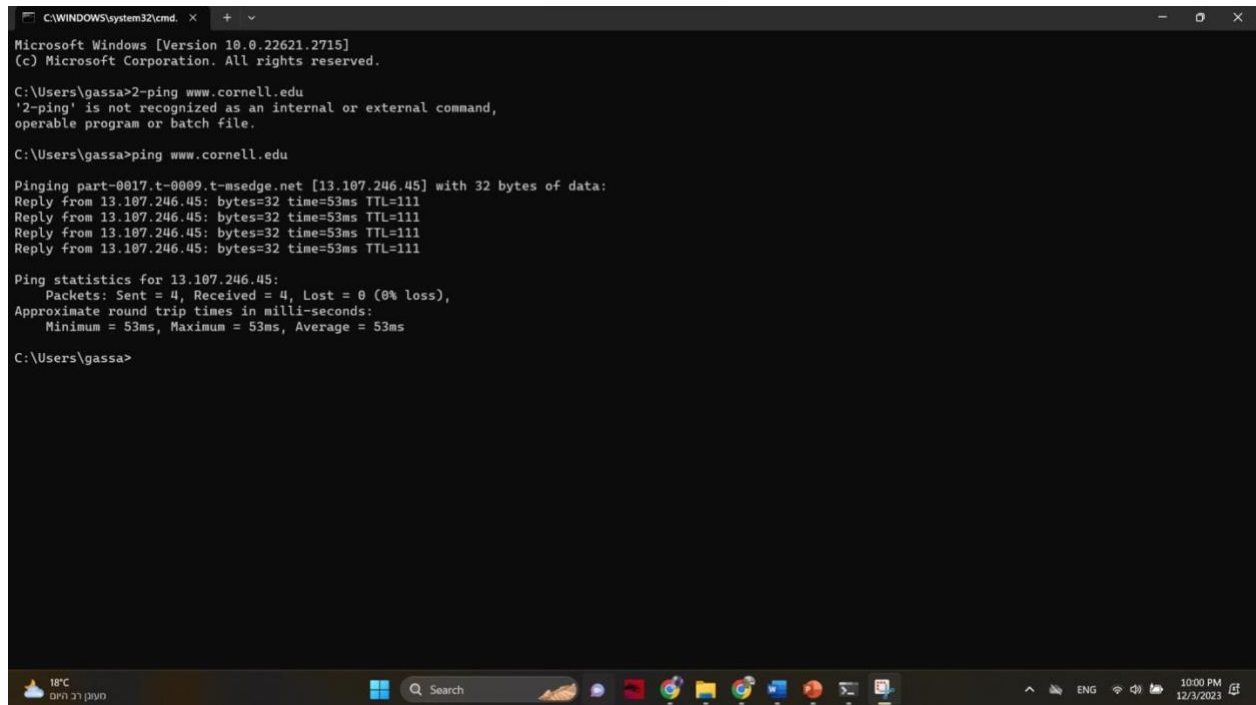
Wireless LAN adapter Wi-Fi:

    Connection-specific DNS Suffix  . : mynet
    Link-local IPv6 Address . . . . . : fe80::bf:6393:f9aa:fe98%4
    IPv4 Address. . . . . : 192.168.1.134
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.1.1

C:\Users\RUZAN>
```

*b) ping [www.cornell.edu](http://www.cornell.edu):*

When you execute the "ping www.cornell.edu" command in the command line, it initiates the ICMP (Internet Control Message Protocol) Echo Request process by sending a sequence of packets to the domain "www.cornell.edu." Initially, the domain is translated into an IP address through DNS resolution. Upon reaching the target server, the packets are returned to your machine, and the time taken for the round-trip is recorded. The output provides information on the number of packets sent, received, and lost, along with statistics on the minimum, maximum, and average round-trip times measured in milliseconds. This utility is valuable for assessing network connectivity and latency between your computer and the Cornell University website.



```
C:\WINDOWS\system32\cmd. x + v
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\gassa>ping www.cornell.edu
'2-ping' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\gassa>ping www.cornell.edu

Pinging part-0017.t-0009.t-msedge.net [13.107.246.45] with 32 bytes of data:
Reply from 13.107.246.45: bytes=32 time=53ms TTL=111
Reply from 13.107.246.45: bytes=32 time=53ms TTL=111
Reply from 13.107.246.45: bytes=32 time=53ms TTL=111
Reply from 13.107.246.45: bytes=32 time=53ms TTL=111

Ping statistics for 13.107.246.45:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 53ms, Maximum = 53ms, Average = 53ms

C:\Users\gassa>
```

c) *tracert* [www.cornell.edu](http://www.cornell.edu):

Executing the "tracert [www.cornell.edu](http://www.cornell.edu)" command in the command line reveals the network path taken by data packets from your computer to the Cornell University website server. This utility systematically sends packets to each hop, typically a router or switch, along the route, documenting the round-trip time for each leg of the journey. The output provides a comprehensive list of intermediate stops (hops) the packets traverse, offering valuable insights for diagnosing potential network issues or identifying bottlenecks.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

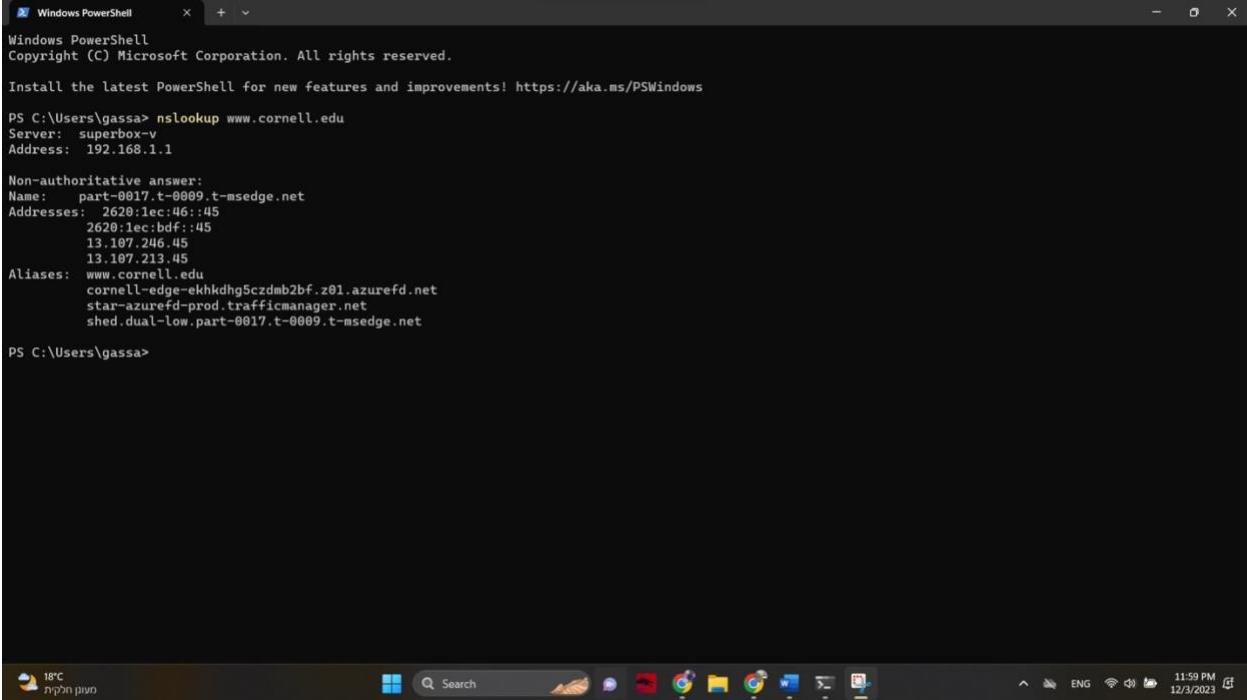
PS C:\Users\gassa> tracert www.cornell.edu

Tracing route to part-0017.t-0009.t-msedge.net [13.107.246.45]
over a maximum of 30 hops:
  0  18 ms  5 ms  2 ms  superbbox-v [192.168.1.1]
  1  21 ms  13 ms  27 ms  10.74.32.250
  2  10 ms  10 ms  12 ms  10.74.22.21
  3  55 ms  53 ms  56 ms  ae61-0.ier02.tlv30.ntwk.msn.net [104.44.36.229]
  4  96 ms  54 ms  54 ms  ae30-0.ear06.mrs20.ntwk.msn.net [104.44.230.199]
  5  54 ms  54 ms  55 ms  be-33-0.ibr01.mrs20.ntwk.msn.net [104.44.33.107]
  6  52 ms  52 ms  52 ms  ae102-0.icr02.mrs20.ntwk.msn.net [104.44.20.56]
  7  *      *      *      Request timed out.
  8  *      *      *      Request timed out.
  9  *      *      *      Request timed out.
 10  *      *      *      Request timed out.
 11  *      *      *      Request timed out.
 12  *      *      *      Request timed out.
 13  *      *      *      Request timed out.
 14  53 ms  52 ms  53 ms  13.107.246.45

Trace complete.
PS C:\Users\gassa>
```

d) *nslookup* [www.cornell.edu](http://www.cornell.edu):

Executing the "nslookup www.cornell.edu" command in the command line involves querying a Domain Name System (DNS) server to translate the domain name www.cornell.edu into its corresponding IP address. This utility furnishes details about the DNS records linked with the domain, particularly the IP address to which the domain name is mapped. The tool is instrumental for troubleshooting DNS-related problems, confirming domain configurations, or obtaining insights into domain addresses.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

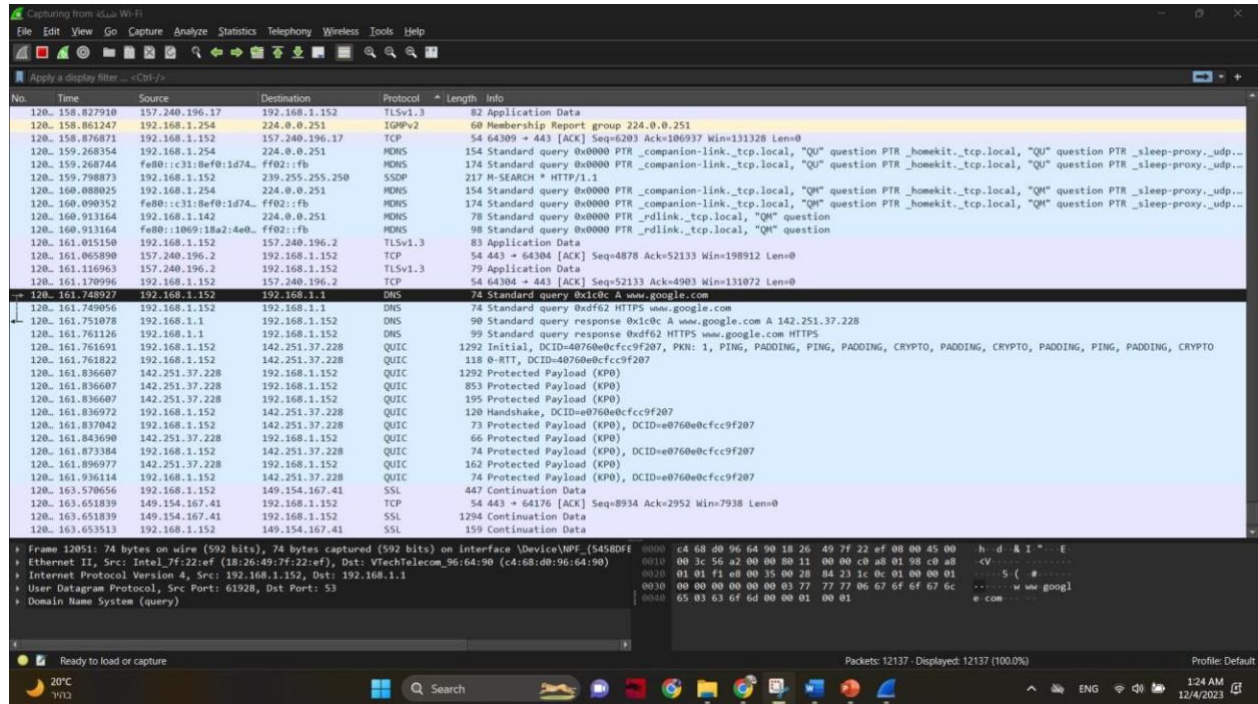
PS C:\Users\gassa> nslookup www.cornell.edu
Server: superbox-v
Address: 192.168.1.1

Non-authoritative answer:
Name:   part-0017.t-0009.t-msedge.net
Addresses:  2620:1ec:46::45
            2620:1ec:bdf::45
            13.107.246.45
            13.107.213.45
Aliases:  www.cornell.edu
          cornell-edge-ekhkdhg5czdmb2bf.z01.azurefd.net
          star-azurefd-prod.trafficmanager.net
          shed.dual-low.part-0017.t-0009.t-msedge.net

PS C:\Users\gassa>
```

### 3) use wireshark to capture some DNS messages:

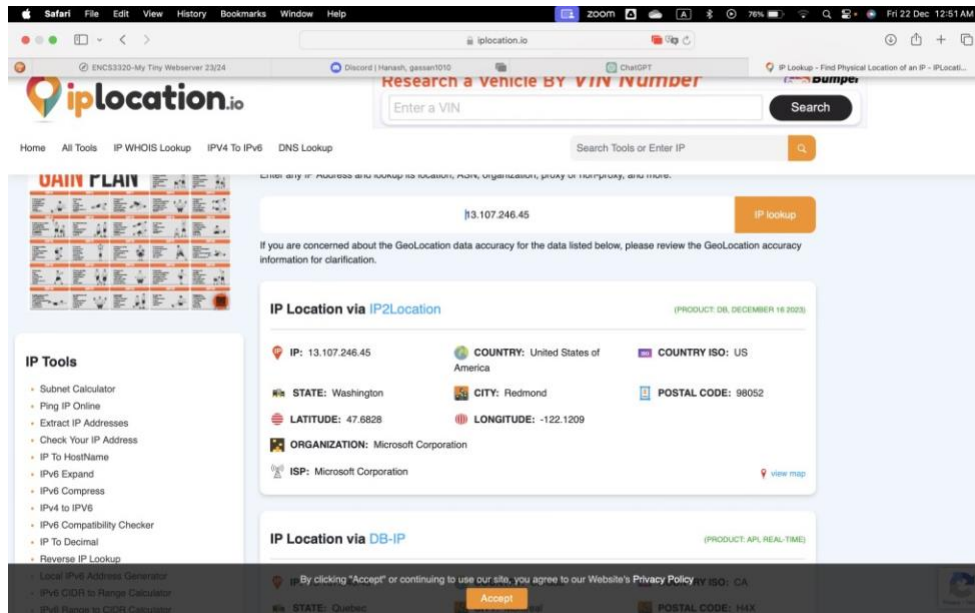
Once you've captured DNS messages, you can analyze the packets to see DNS query and response details. Look for DNS packets and inspect the query and response sections. Regarding the ping results and determining if the response is from the USA, ping response times alone are not reliable indicators of geographical location. While longer response times might suggest greater distance.





- From the ping results, do you think the response you have got is from USA?  
Explain your answer briefly.

After looking up for the IP address using “https://iplocation.io/ip/13.107.246.45” website that we got after applying “ping [www.cornell.edu](http://www.cornell.edu)” command in cmd line window, here is the result:

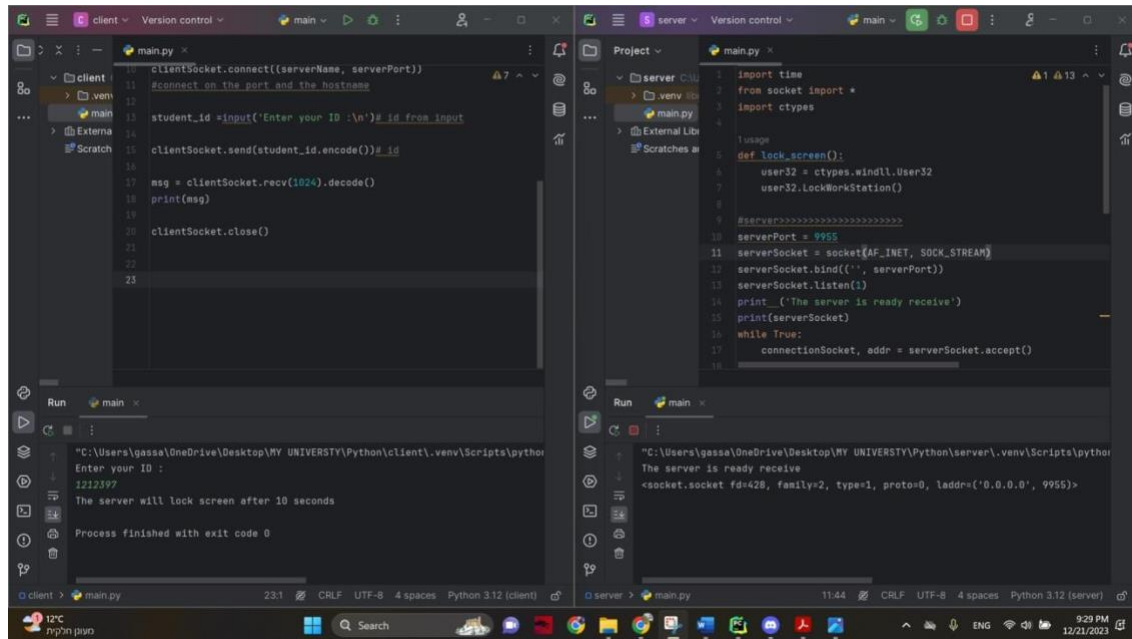


As we can see the response is from USA.

## Part2: socket programing using python:

In this part we have used socket programing to implement TCP client and server applications using python, so in this program the client sends a message to the server if the message contains any of our ID's it will display a message that the screen will lock after 10 seconds and lock it after 10 seconds, if it's not it will display an error message on the server side without locking the screen.

Those are 3 successful runs with our 3 different ID's:



```
client/main.py
1 clientSocket.connect((serverName, serverPort))
2 #connect on the port and the hostname
3
4 student_id = input('Enter your ID :\n') # id from input
5
6 clientSocket.send(student_id.encode()) # id
7
8 msg = clientSocket.recv(1024).decode()
9 print(msg)
10
11 clientSocket.close()
12
13
14
15
16
17
18
19
20
21
22
23

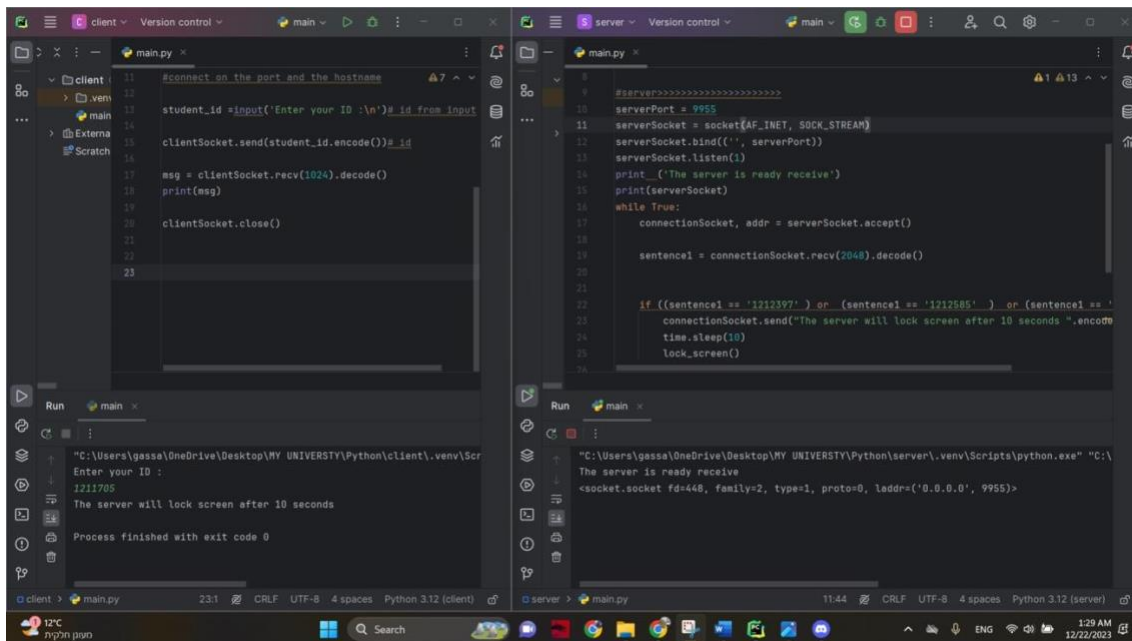
server/main.py
1 import time
2 from socket import *
3 import ctypes
4
5 #usage
6 def lock_screen():
7     user32 = ctypes.windll.User32
8     user32.LockWorkStation()
9
10 #server
11 serverPort = 9955
12 serverSocket = socket(AF_INET, SOCK_STREAM)
13 serverSocket.bind(('', serverPort))
14 serverSocket.listen(1)
15 print('The server is ready receive')
16 print(serverSocket)
17 while True:
18     connectionSocket, addr = serverSocket.accept()
19
```

Run client main.py

```
"C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\client\.venv\Scripts\python.exe" "C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\client\main.py"
Enter your ID :
1212397
The server will lock screen after 10 seconds
Process finished with exit code 0
```

Run server main.py

```
"C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\server\.venv\Scripts\python.exe" "C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\server\main.py"
The server is ready receive
<socket.socket fd=428, family=2, type=1, proto=0, laddr=('0.0.0.0', 9955)>
```



```
client/main.py
1 #connect on the port and the hostname
2
3 student_id = input('Enter your ID :\n') # id from input
4
5 clientSocket.send(student_id.encode()) # id
6
7 msg = clientSocket.recv(1024).decode()
8 print(msg)
9
10 clientSocket.close()
11
12
13
14
15
16
17
18
19
20
21
22
23

server/main.py
1
2 #server
3 serverPort = 9955
4 serverSocket = socket(AF_INET, SOCK_STREAM)
5 serverSocket.bind(('', serverPort))
6 serverSocket.listen(1)
7 print('The server is ready receive')
8 print(serverSocket)
9 while True:
10     connectionSocket, addr = serverSocket.accept()
11
12     sentence1 = connectionSocket.recv(2048).decode()
13
14     if ((sentence1 == '1212397') or (sentence1 == '1212585') or (sentence1 == '1211705')):
15         connectionSocket.send("The server will lock screen after 10 seconds ".encode())
16         time.sleep(10)
17         lock_screen()
18
```

Run client main.py

```
"C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\client\.venv\Scripts\python.exe" "C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\client\main.py"
Enter your ID :
1211705
The server will lock screen after 10 seconds
Process finished with exit code 0
```

Run server main.py

```
"C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\server\.venv\Scripts\python.exe" "C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\server\main.py"
The server is ready receive
<socket.socket fd=448, family=2, type=1, proto=0, laddr=('0.0.0.0', 9955)>
```

```

#client
11 #connect on the port and the hostname
12
13 student_id = input('Enter your ID :\n')# id from input
14
15 clientSocket.send(student_id.encode())# id
16
17 msg = clientSocket.recv(1024).decode()
18 print(msg)
19
20 clientSocket.close()
21
22
23

```

```

#server
9 #server
10 serverPort = 9955
11 serverSocket = socket(AF_INET, SOCK_STREAM)
12 serverSocket.bind(('', serverPort))
13 serverSocket.listen(1)
14 print('The server is ready receive')
15 print(serverSocket)
16 while True:
17     connectionSocket, addr = serverSocket.accept()
18
19     sentence1 = connectionSocket.recv(2048).decode()
20
21
22     if ((sentence1 == '1212397') or (sentence1 == '1212585') or (sentence1 == '
23 connectionSocket.send("The server will lock screen after 10 seconds ".encode
24 time.sleep(10)
25 lock_screen()
26

```

Run client: "C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\client\venv\Scr Enter your ID : 1212585 The server will lock screen after 10 seconds Process finished with exit code 0

Run server: "C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\server\venv\Scripts\python.exe" "C:\ The server is ready receive <socket.socket fd=448, family=2, type=1, proto=0, laddr=('0.0.0.0', 9955)>

And this is a run gives an error because we didn't entered on of our ID's:

```

#client
11 #connect on the port and the hostname
12
13 student_id = input('Enter your ID :\n')# id from input
14
15 clientSocket.send(student_id.encode())# id
16
17 msg = clientSocket.recv(1024).decode()
18 print(msg)
19
20 clientSocket.close()
21
22
23

```

```

#server
9 #server
10 serverPort = 9955
11 serverSocket = socket(AF_INET, SOCK_STREAM)
12 serverSocket.bind(('', serverPort))
13 serverSocket.listen(1)
14 print('The server is ready receive')
15 print(serverSocket)
16 while True:
17     connectionSocket, addr = serverSocket.accept()
18
19     sentence1 = connectionSocket.recv(2048).decode()
20
21
22     if ((sentence1 == '1212397') or (sentence1 == '1212585') or (sentence1 == '
23 connectionSocket.send("The server will lock screen after 10 seconds ".encode
24 time.sleep(10)
25 lock_screen()
26

```

Run client: "C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\client\venv\Scr Enter your ID : 222222222222

Run server: "C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\server\venv\Scripts\python.exe" "C:\ The server is ready receive <socket.socket fd=448, family=2, type=1, proto=0, laddr=('0.0.0.0', 9955)> error : invalid text

```

#client from socket
import *
serverName =
'localhost'
serverPort =

```

```
9955 #define name and the
port

#client >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

clientSocket = socket(AF_INET, SOCK_STREAM)
# create the TCP client server
clientSocket.connect((serverName, serverPort))
#connect on the port and the hostname

student_id=input('Enter your ID :\n')# id from input

clientSocket.send(student_id.encode())# id

msg = clientSocket.recv(1024).decode()
print(msg)
clientSocket.close()


#server
import time from
socket import *
import ctypes
def
lock_screen():
    user32 = ctypes.windll.User32
    user32.LockWorkStation()
#server>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> serverPort
= 9955
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind('', serverPort)) serverSocket.listen(1)
print ('The server is ready receive')
print(serverSocket) while True:
    connectionSocket, addr = serverSocket.accept()

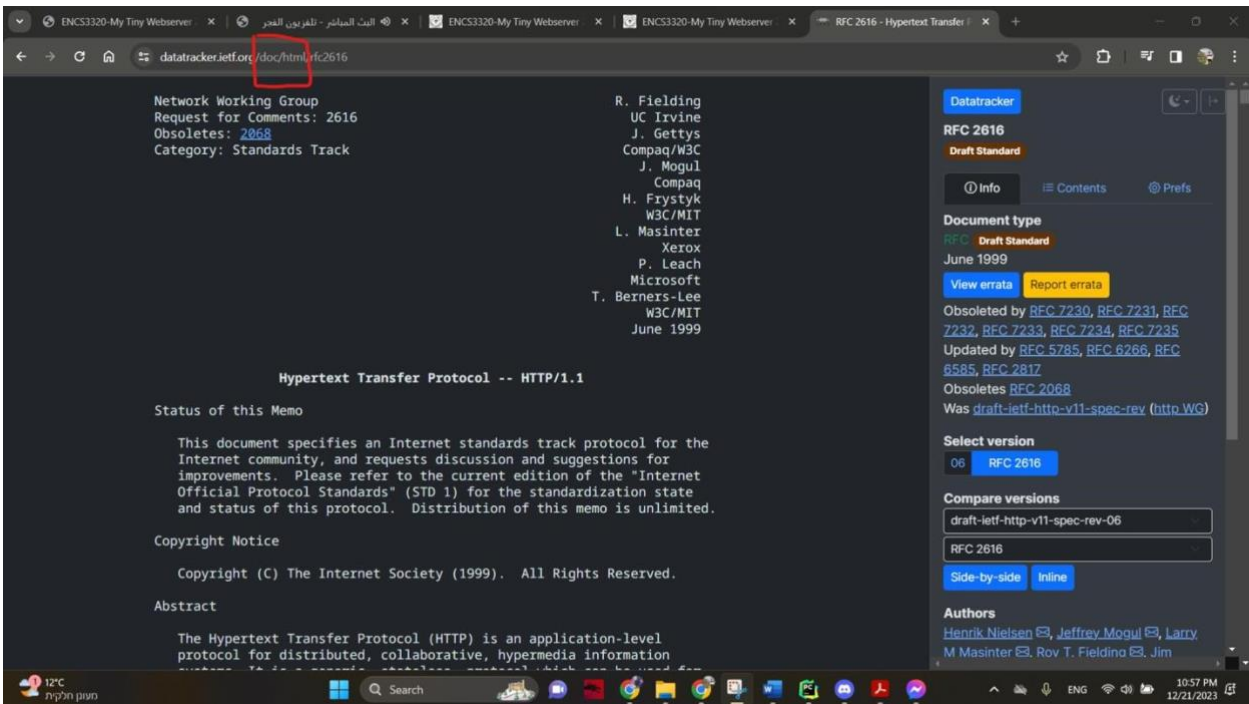
    sentencel = connectionSocket.recv(2048).decode()
        if ((sentencel == '1212397' ) or (sentencel == '1212585' ))
or
(sentencel == '1211705' )):
            connectionSocket.send("The
server will lock screen after 10 seconds ".encode())
time.sleep(10)
            lock_screen()

else:
    print(" error : invalid text")
connectionSocket.close()
```

### part3:

0) from rfce2616, what is Content-Type in the HTTP request and why do we need it?

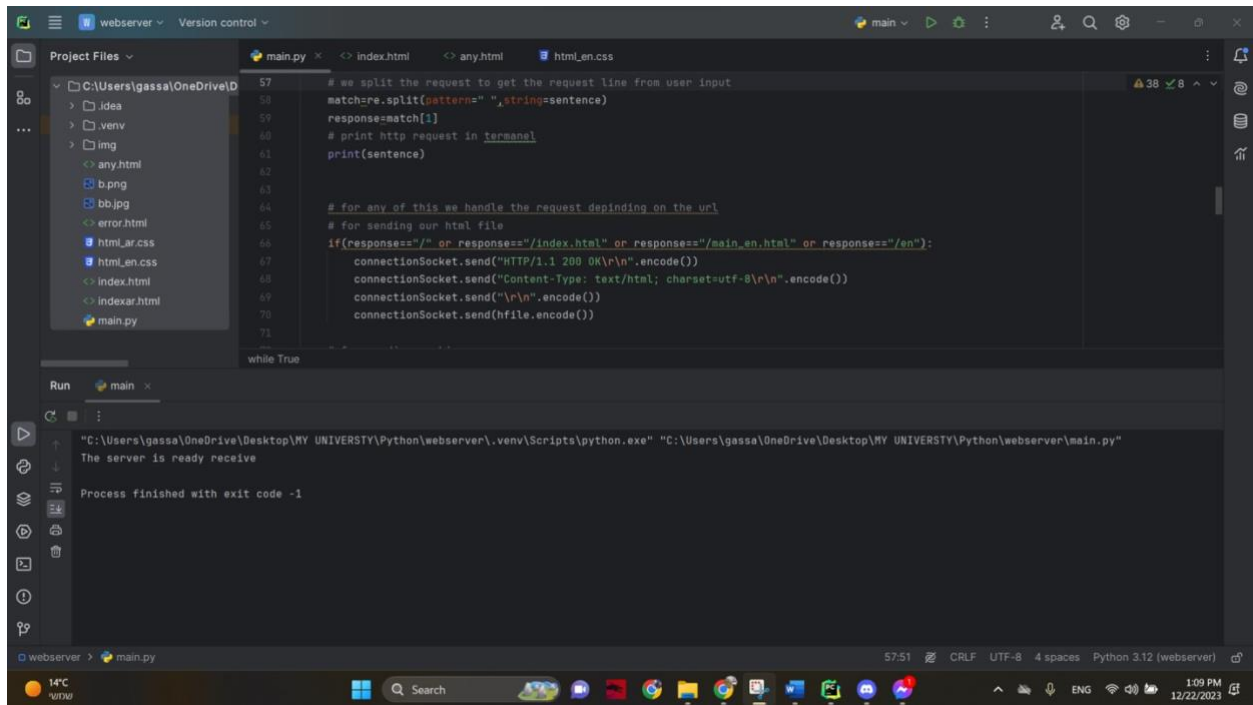
The Content-Type entity-header field indicates the media type of the entity-body sent to the recipient or, in the case of the HEAD method, the media type that would have been sent had the request been a GET, and we need it Because it tells the server what kind of data is being sent in the request so that the server can properly interpret and process it, it is crucial for both the client and server to understand how to handle the data being exchanged.



The screenshot shows a web browser displaying the RFC 2616 page on datacenter.ietf.org. The URL bar shows 'datacenter.ietf.org/doc/html/rfc2616'. The page content includes the title 'Hypertext Transfer Protocol -- HTTP/1.1', a list of authors (R. Fielding, UC Irvine, J. Gettys, Compaq/W3C, J. Mogul, Compaq, H. Frystyk, W3C/MIT, L. Masinter, Xerox, P. Leach, Microsoft, T. Berners-Lee, W3C/MIT, June 1999), and a 'Status of this Memo' section. The 'Status of this Memo' section states: 'This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.' The 'Copyright Notice' section states: 'Copyright (C) The Internet Society (1999). All Rights Reserved.' The 'Abstract' section states: 'The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It has been created by the W3C and is intended to be the core protocol for the World Wide Web.' The right sidebar shows the 'Datatracker' for RFC 2616, 'Draft Standard', with links for 'Info', 'Contents', and 'Prefs'. It also includes a 'Document type' section, 'View errata', 'Report errata', and a 'Select version' dropdown menu.

1) if the request is / or /index.html or /main\_en.html or /en (for example localhost:9966/ or localhost:9966/en) then the server should send main\_en.html file with Content-Type: text/html:

in this part if the request was any one of the above requests so the server send the English html web bage



The screenshot shows a code editor with a project named 'webserver'. The file explorer on the left lists files: main.py, index.html, any.html, html\_en.css, error.html, b.png, bb.jpg, html\_ar.css, index.html, indexar.html, and main.py. The main.py file is open, showing Python code for a web server. The code includes comments and logic to handle requests for '/', '/index.html', '/main\_en.html', and '/en'. It uses the 'http.server' module and 'urllib.parse' to parse the request. The code sends an HTTP 200 OK response and the content of the requested file. The terminal at the bottom shows the command to run the server: 'python main.py' and the output: 'The server is ready receive'. The status bar at the bottom indicates the file encoding is UTF-8, the line ending is CRLF, and the Python version is 3.12.

```
57 # we split the request to get the request line from user input
58 match=re.split(pattern=" ",string=sentence)
59 response=match[1]
60 # print http request in terminal
61 print(sentence)
62
63
64 # for any of this we handle the request depending on the url
65 # for sending our html file
66 if(response=="/" or response=="/index.html" or response=="/main_en.html" or response=="/en"):
67     connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
68     connectionSocket.send("Content-Type: text/html; charset=utf-8\r\n".encode())
69     connectionSocket.send("\r\n".encode())
70     connectionSocket.send(hfile.encode())
71
```

Run main x

"C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\webserver\.venv\Scripts\python.exe" "C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\webserver\main.py"

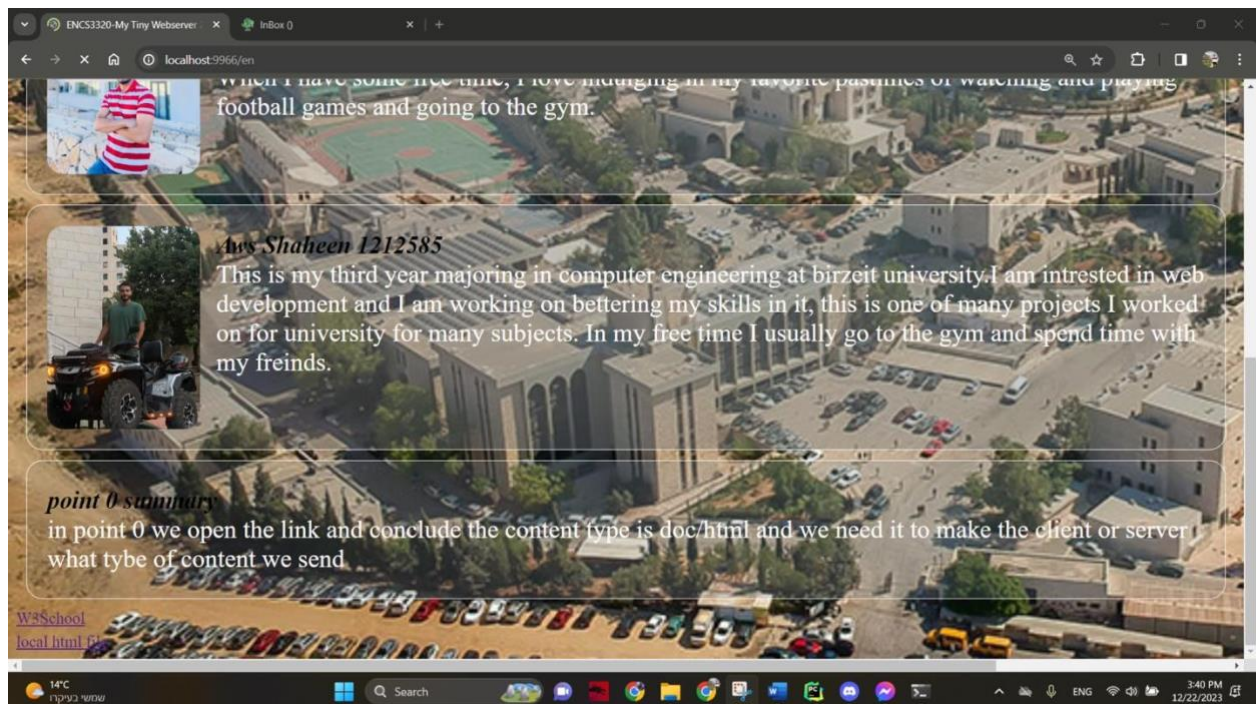
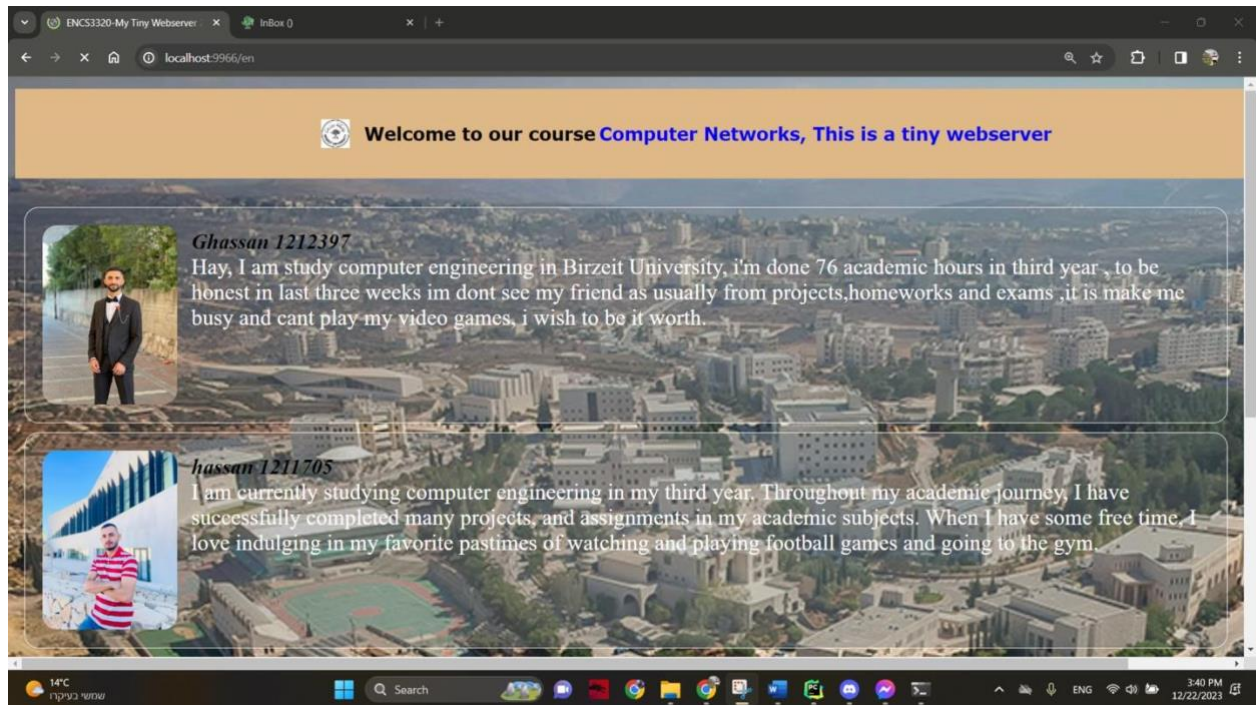
The server is ready receive

Process finished with exit code -1

webserver > main.py 57:51 CRLF UTF-8 4 spaces Python 3.12 (webserver)

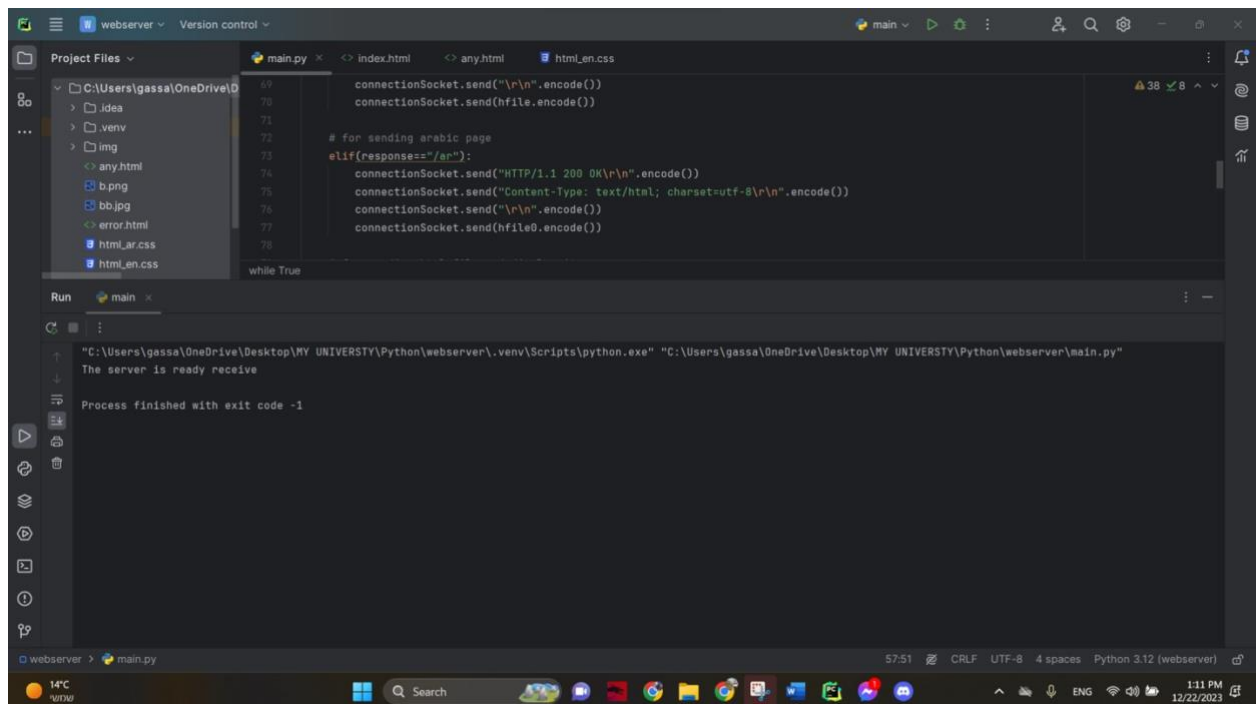


This is the response from server, it is our website bage:



2) If the request is /ar then the server response with main\_ar.html which is an Arabic version of main\_en.html:

In this part if the request was /ar then the response from server should be the Arabic html web page



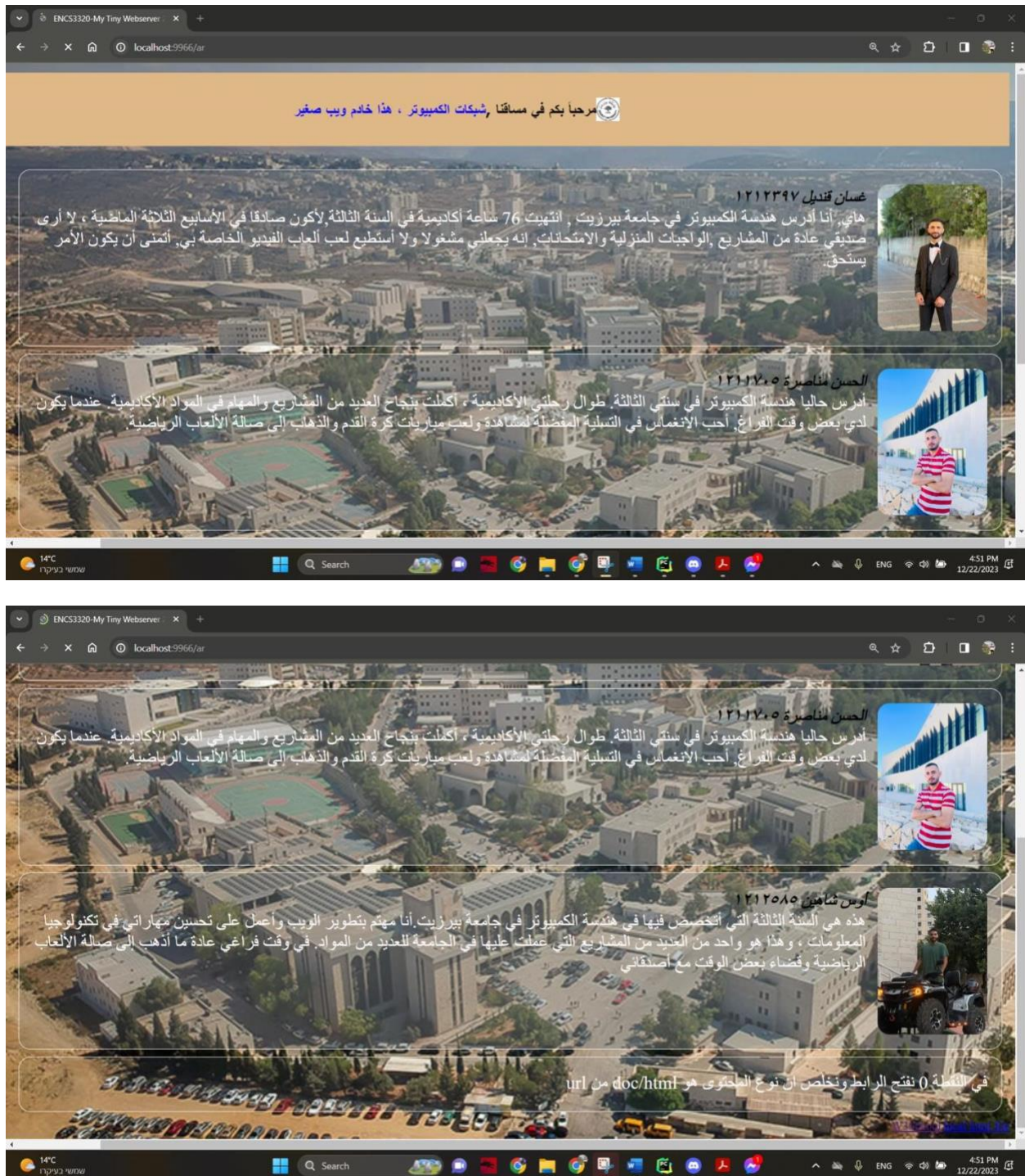
```
69 connectionSocket.send("\r\n".encode())
70 connectionSocket.send(hfile.encode())
71
72 # for sending arabic page
73 elif(response=="ar"):
74     connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
75     connectionSocket.send("Content-Type: text/html; charset=utf-8\r\n".encode())
76     connectionSocket.send("\r\n".encode())
77     connectionSocket.send(hfile0.encode())
78
79 while True
```

Run main

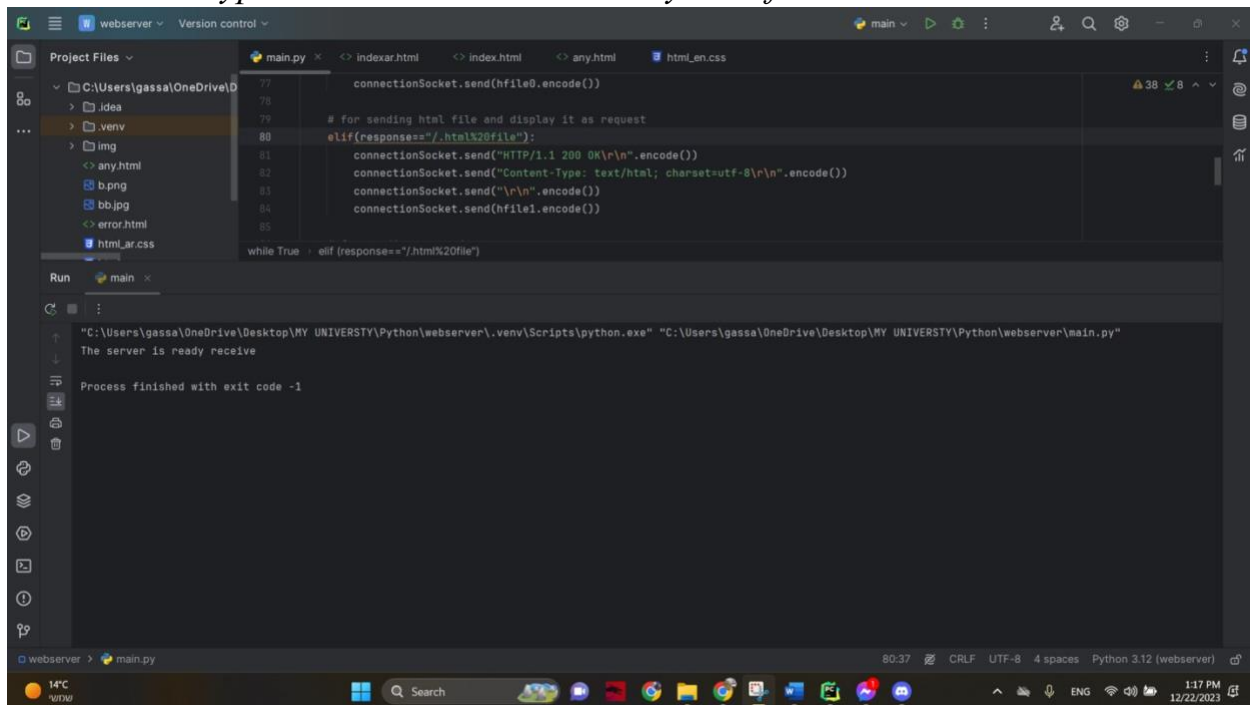
```
"C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\webserver\.venv\Scripts\python.exe" "C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\webserver\main.py"
The server is ready receive
Process finished with exit code -1
```



This is the response we got from server it is our website page :



3) if the request is an .html file then the server should send the requested html file with Content-Type: text/html. You can use any html file:

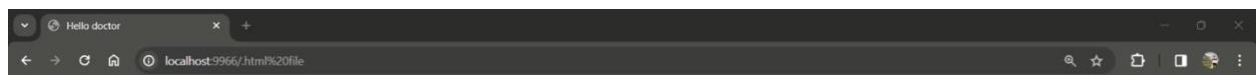


The screenshot shows an IDE window with a project named 'webserver'. The file explorer on the left shows a directory structure with files like 'any.html', 'b.png', 'bb.jpg', 'error.html', and 'html\_ar.css'. The main editor displays a Python script 'main.py' with the following code:

```
77 connectionSocket.send(hfile0.encode())
78
79 # for sending html file and display it as request
80 elif(response=="/.html%20file"):
81     connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
82     connectionSocket.send("Content-Type: text/html; charset=utf-8\r\n".encode())
83     connectionSocket.send("\r\n".encode())
84     connectionSocket.send(hfile1.encode())
85
while True: elif (response=="/.html%20file")
```

The 'Run' console at the bottom shows the command executed: `"C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\webserver\.venv\Scripts\python.exe" "C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\webserver\main.py"`. The output indicates the server is ready to receive and that the process finished with exit code -1.

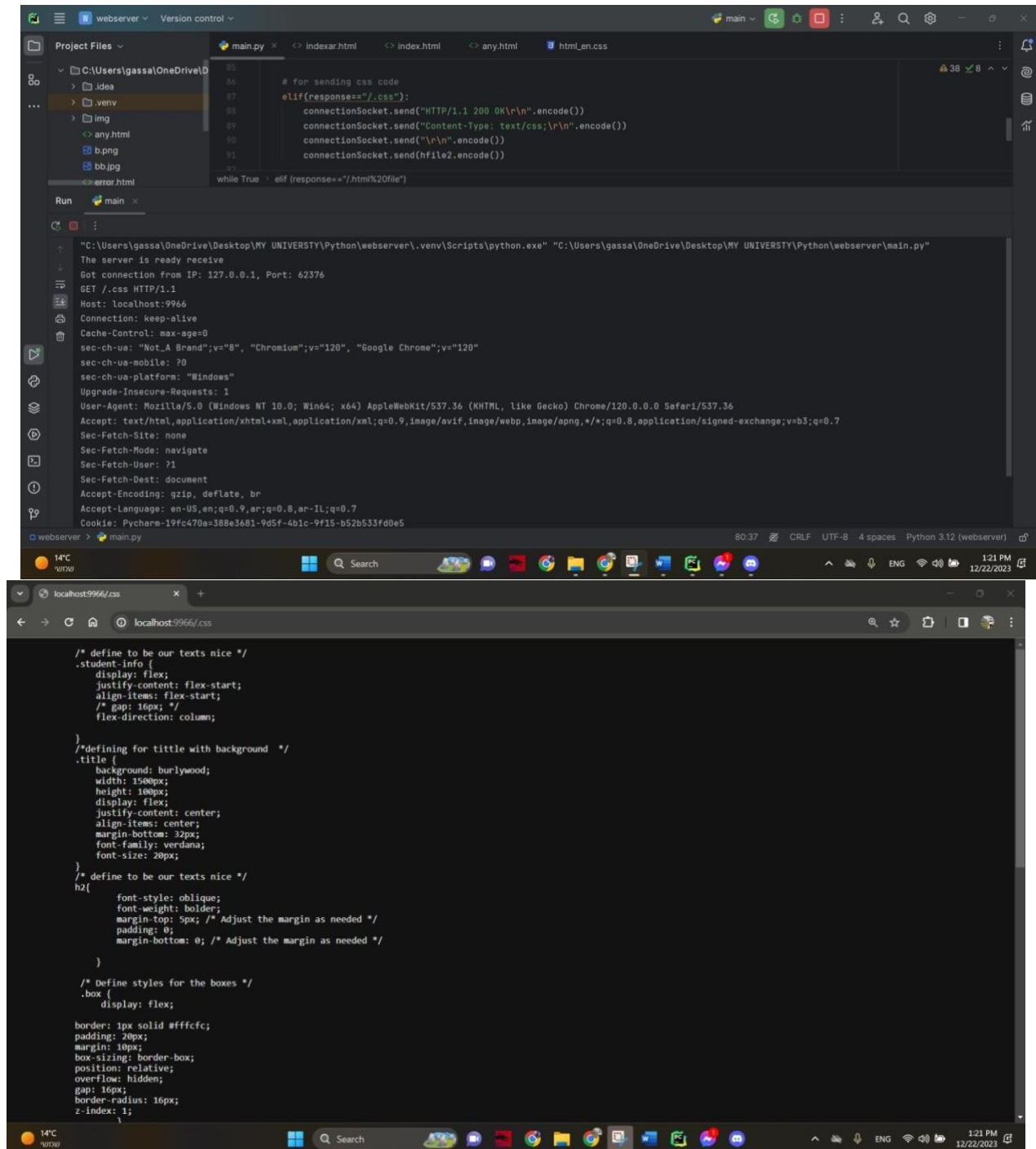
The response is this simple html server:



Hope this message find you well Doctor, and we wish GAZA be ok



4) if the request is a `.css` file then the server should send the requested css file with `Content-Type: text/css`. You can use any CSS file:



The image shows a web server running in a terminal window and a browser window displaying a CSS file.

**Terminal Window (Top):**

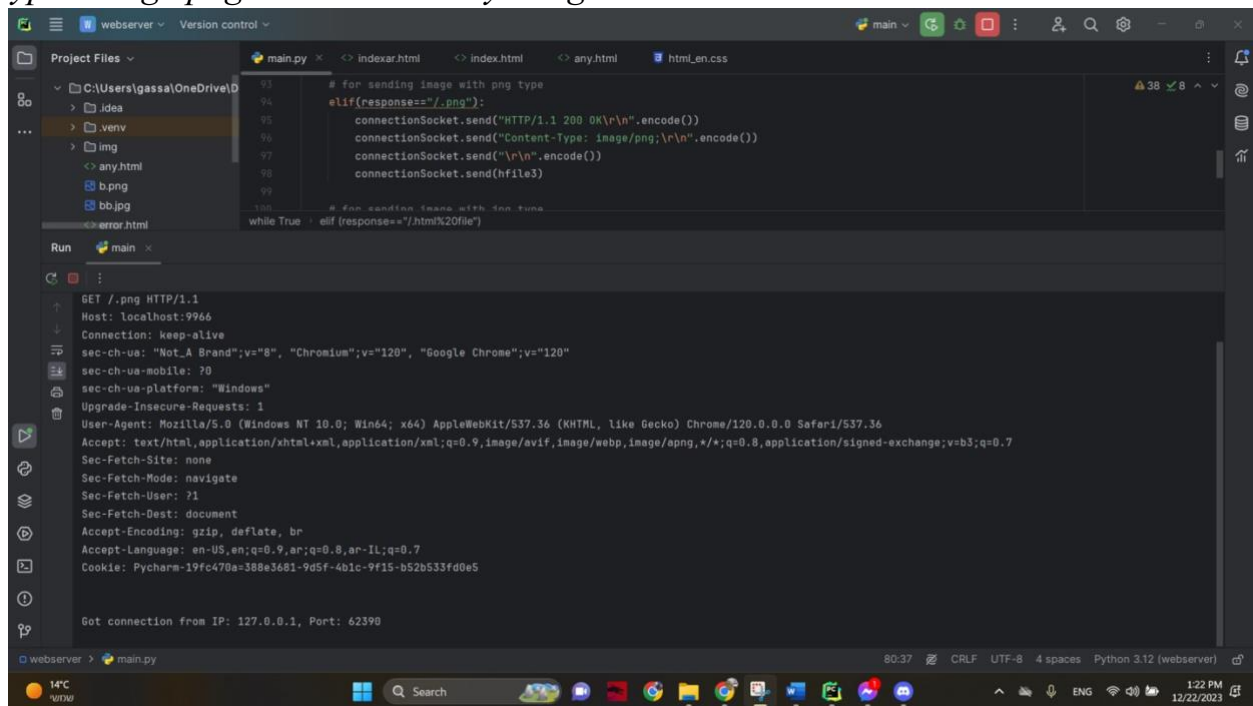
```
main.py
# for sending css code
elif(response=="/.css"):
    connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
    connectionSocket.send("Content-Type: text/css;\r\n".encode())
    connectionSocket.send("\r\n".encode())
    connectionSocket.send(hfile2.encode())
while True:
    elif (response=="html%20file"):
```

**Browser Window (Bottom):**

```
localhost:9966/css
/* define to be our texts nice */
.student-info {
    display: flex;
    justify-content: flex-start;
    align-items: flex-start;
    /* gap: 16px; */
    flex-direction: column;
}
/*defining for tittle with background */
.title {
    background: burlywood;
    width: 1500px;
    height: 100px;
    display: flex;
    justify-content: center;
    align-items: center;
    margin-bottom: 32px;
    font-family: verdana;
    font-size: 20px;
}
/* define to be our texts nice */
h2{
    font-style: oblique;
    font-weight: bolder;
    margin-top: 5px; /* Adjust the margin as needed */
    padding: 0;
    margin-bottom: 0; /* Adjust the margin as needed */
}
/* Define styles for the boxes */
.box {
    display: flex;
    border: 1px solid #ffffcc;
    padding: 20px;
    margin: 10px;
    box-sizing: border-box;
    position: relative;
    overflow: hidden;
    gap: 16px;
    border-radius: 16px;
    z-index: 1;
}
```



5) if the request is a **.png** then the server should send the png image with Content-Type: image/png. You can use any image

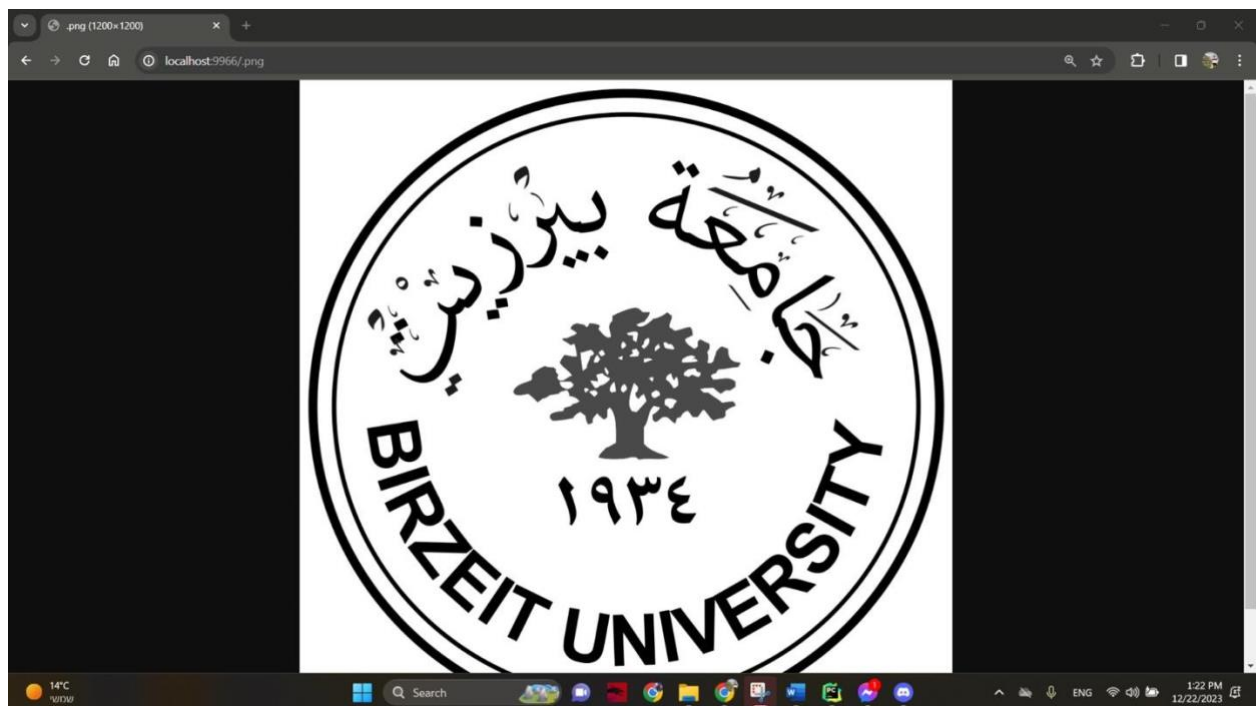


```
Project Files
  C:\Users\gassa\OneDrive\
  > .idea
  > .venv
  > img
  > any.html
  > b.png
  > bb.jpg
  > error.html

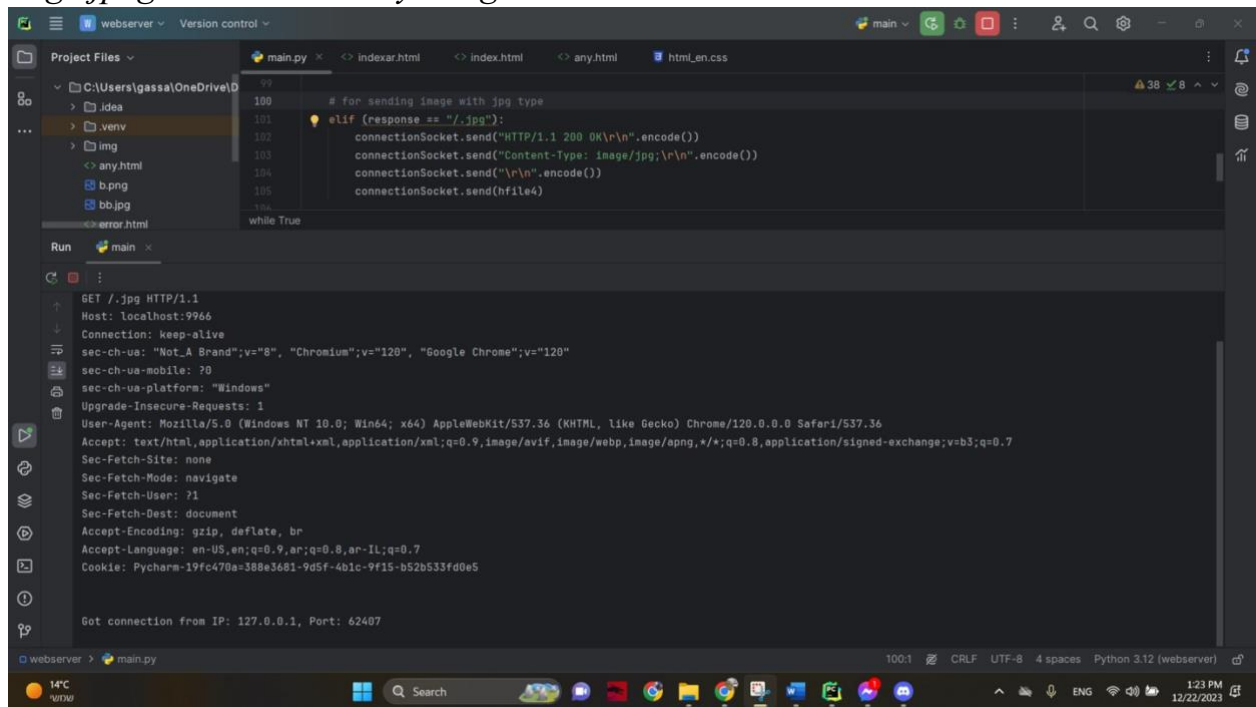
main.py
  93 # for sending image with png type
  94 elif(response=="/.png"):
  95     connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
  96     connectionSocket.send("Content-Type: image/png;\r\n".encode())
  97     connectionSocket.send("\r\n".encode())
  98     connectionSocket.send(hfile3)
  99
  100 # for sending image with any type
  while True:
  elif (response=="./html%20file")
```

```
Run
main
GET /.png HTTP/1.1
Host: localhost:9966
Connection: keep-alive
sec-ch-ua: "Not A Brand";v="8", "Chromium";v="120", "Google Chrome";v="120"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,ar;q=0.8,ar-IL;q=0.7
Cookie: Pycharm-19fc470a-388e3681-9d5f-4b1c-9f15-b52b533fd0e5

Got connection from IP: 127.0.0.1, Port: 62390
```



6) if the request is a **.jpg** then the server should send the jpg image with *ContentType: image/jpeg*. You can use any image:



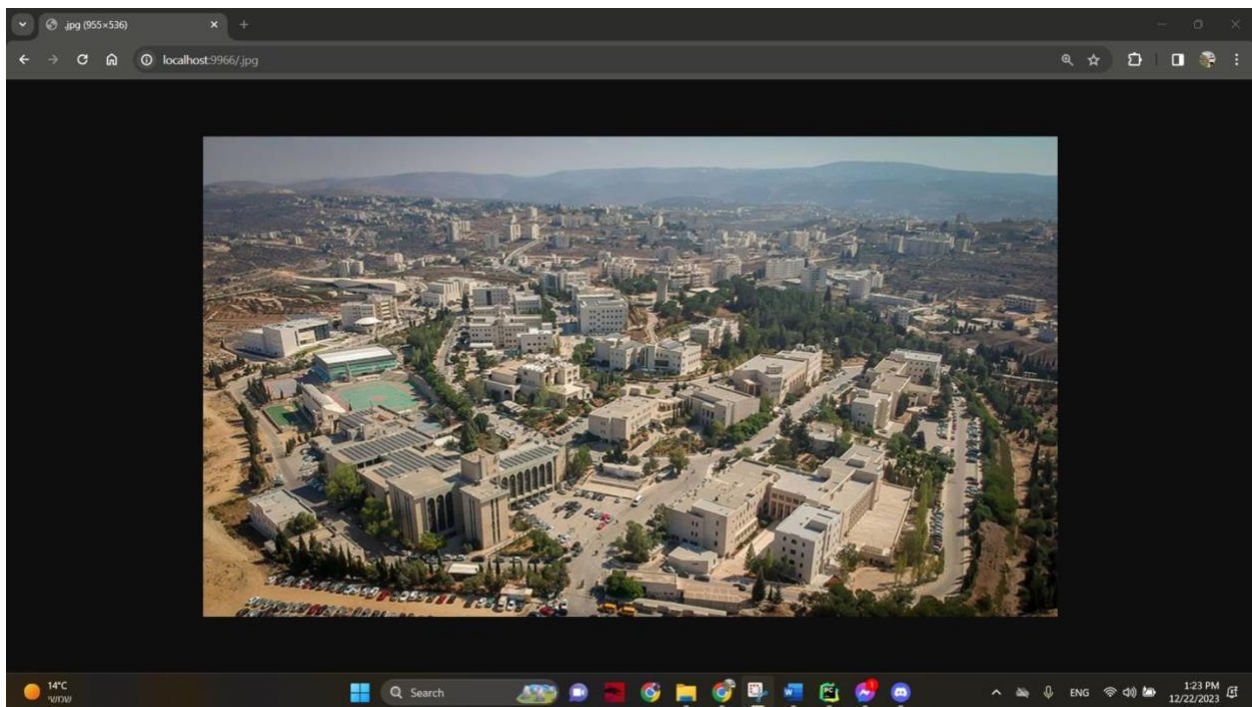
The screenshot shows the PyCharm IDE with a Python web server project. The code in `main.py` is as follows:

```
99
100 # for sending image with jpg type
101 elif (response == "/.jpg"):
102     connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
103     connectionSocket.send("Content-Type: image/jpeg;\r\n".encode())
104     connectionSocket.send("\r\n".encode())
105     connectionSocket.send(hfile4)
106 while True:
```

The Run console shows the following output:

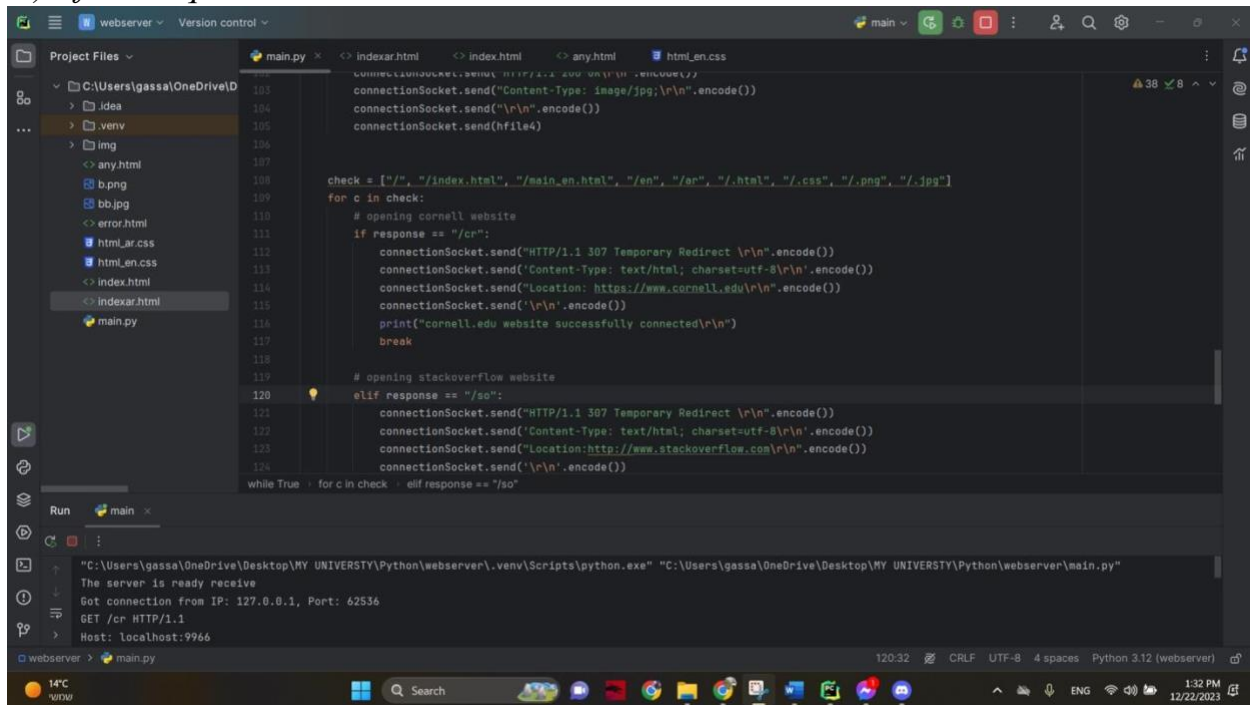
```
GET /.jpg HTTP/1.1
Host: localhost:9966
Connection: keep-alive
sec-ch-ua: "Not_A Brand";v="8", "Chromium";v="120", "Google Chrome";v="120"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,ar;q=0.8,he-IL;q=0.7
Cookie: Pycharm-19fc470a-388e3681-9d5f-4b1c-9f15-b52b533fd0e5

Got connection from IP: 127.0.0.1, Port: 62407
```



7) Use the status code 307 Temporary Redirect to redirect the following

a) If the request is /cr then redirect to cornell.edu website:



The screenshot shows a VS Code editor with a Python file named `main.py`. The script uses `http.server` to serve files from a directory. It includes a check for specific file extensions and a redirect logic for the `/cr` path. The terminal output shows the server running on `localhost:9666` and receiving a `GET /cr HTTP/1.1` request.

```
103 connectionSocket.send('HTTP/1.1 200 OK\r\n\r\n')
104 connectionSocket.send('Content-Type: image/jpg;\r\n\r\n'.encode())
105 connectionSocket.send(hfile4)
106 connectionSocket.send(hfile4)
107
108 check = ["/", "/index.html", "/main_en.html", "/en", "/ar", "/f.html", "/f.css", "/f.png", "/f.jpg"]
109 for c in check:
110     # opening cornell website
111     if response == "/cr":
112         connectionSocket.send('HTTP/1.1 307 Temporary Redirect \r\n\r\n'.encode())
113         connectionSocket.send('Content-Type: text/html; charset=utf-8\r\n\r\n'.encode())
114         connectionSocket.send('Location: https://www.cornell.edu\r\n\r\n'.encode())
115         connectionSocket.send('\r\n\r\n'.encode())
116         print("cornell.edu website successfully connected\r\n")
117         break
118
119 # opening stackoverflow website
120 elif response == "/so":
121     connectionSocket.send('HTTP/1.1 307 Temporary Redirect \r\n\r\n'.encode())
122     connectionSocket.send('Content-Type: text/html; charset=utf-8\r\n\r\n'.encode())
123     connectionSocket.send('Location: http://www.stackoverflow.com\r\n\r\n'.encode())
124     connectionSocket.send('\r\n\r\n'.encode())
125
126 while True:
127     for c in check:
128         if response == "/so":
```

Run main.py

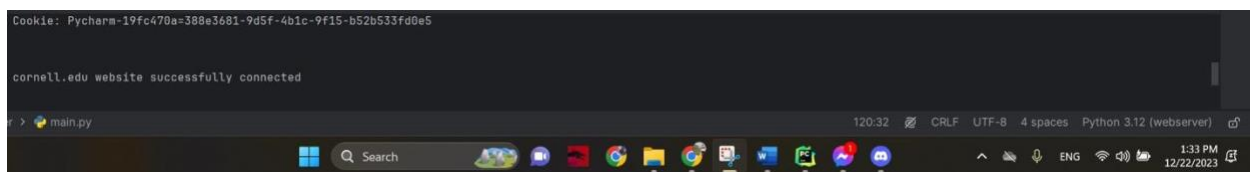
"C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\webserver\venv\Scripts\python.exe" "C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\webserver\main.py"

The server is ready receive

Got connection from IP: 127.0.0.1, Port: 62536

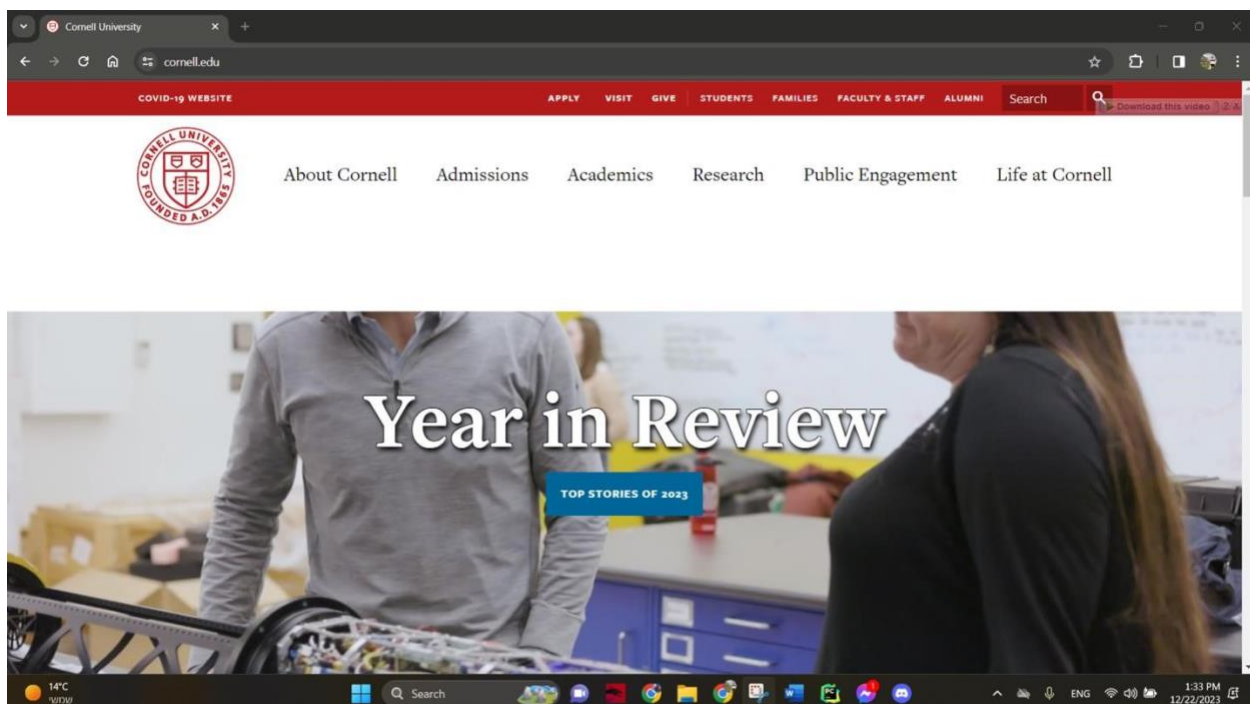
GET /cr HTTP/1.1

Host: localhost:9666

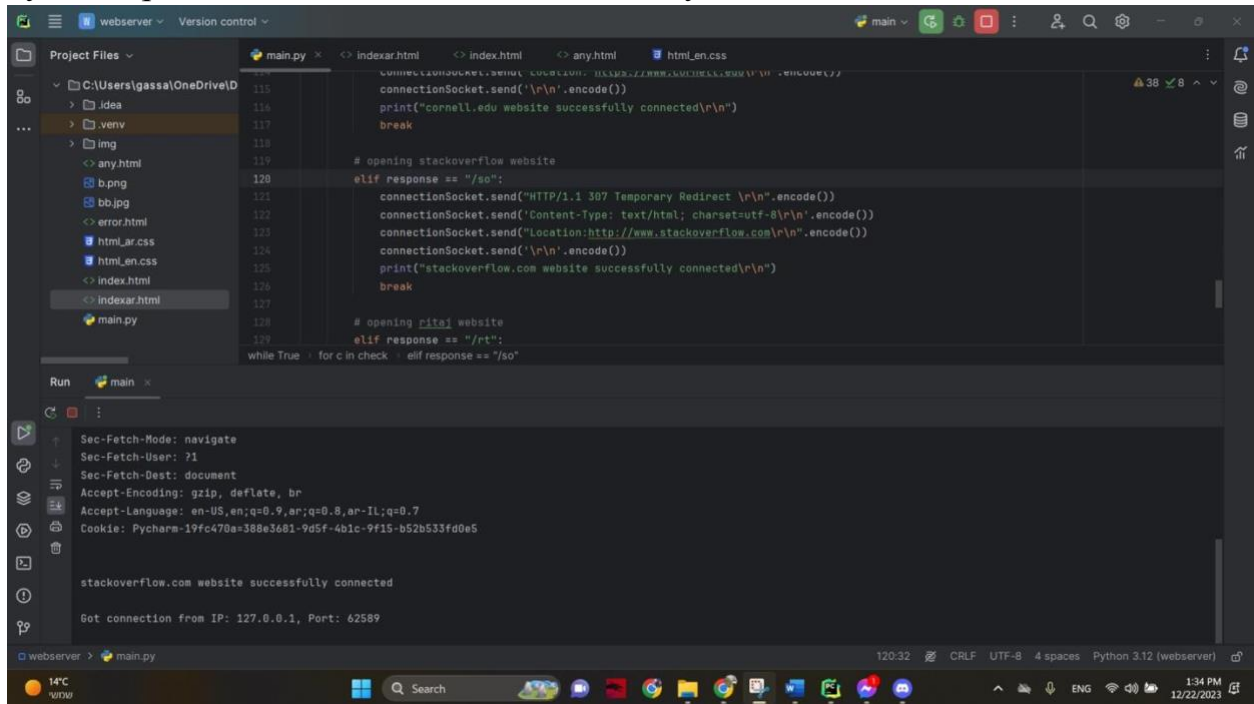


The terminal window shows the output of the script, including the cookie `Pycharm-19fc470a-308e3681-9d5f-4b1c-9f15-b52b533fd0e5` and the message `cornell.edu website successfully connected`.

```
Cookie: Pycharm-19fc470a-308e3681-9d5f-4b1c-9f15-b52b533fd0e5
cornell.edu website successfully connected
```



b) If the request is /so then redirect to stackoverflow.com website:



The screenshot shows a code editor with a project named 'webserver'. The file explorer on the left shows a directory structure with files like 'any.html', 'b.png', 'bb.jpg', 'error.html', 'html\_ar.css', 'html\_en.css', 'index.html', and 'main.py'. The main editor displays the code for 'main.py', which is a Python script using the 'socket' module to handle HTTP requests. The code includes logic to redirect requests for '/so' to 'http://www.stackoverflow.com'. The 'Run' panel at the bottom shows the output of the server, including headers like 'Sec-Fetch-Mode: navigate', 'Sec-Fetch-User: ?1', 'Sec-Fetch-Dest: document', 'Accept-Encoding: gzip, deflate, br', 'Accept-Language: en-US,en;q=0.9,ar;q=0.8,ar-IL;q=0.7', and 'Cookie: Pycharm-19fc470a-388e3681-9d5f-4b1c-9f15-b52b533fd0e5'. The output also shows 'stackoverflow.com website successfully connected' and 'Got connection from IP: 127.0.0.1, Port: 62589'.

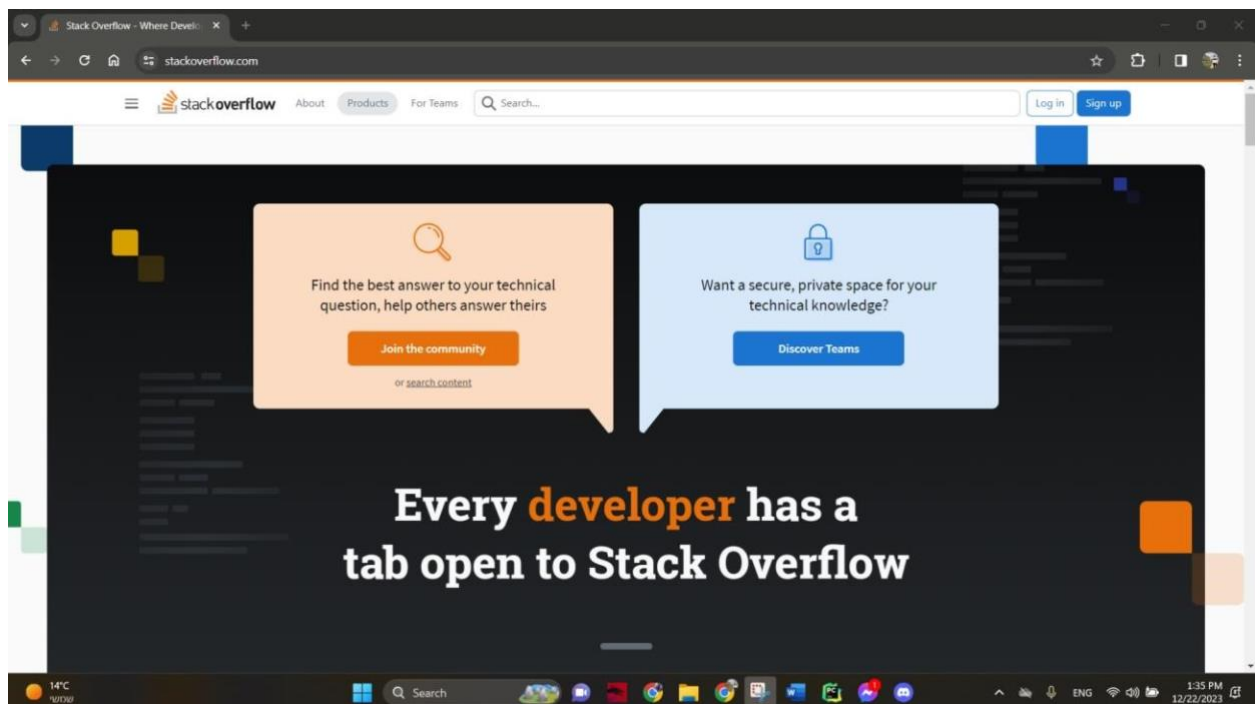
```
main.py
115 connectionSocket.send('HTTP/1.1 307 Temporary Redirect \r\n'.encode())
116 connectionSocket.send('Content-type: text/html; charset=utf-8\r\n'.encode())
117 print("cornell.edu website successfully connected\r\n")
118 break
119
120 # opening stackoverflow website
121 elif response == "/so":
122     connectionSocket.send("HTTP/1.1 307 Temporary Redirect \r\n".encode())
123     connectionSocket.send('Content-type: text/html; charset=utf-8\r\n'.encode())
124     connectionSocket.send('Location:http://www.stackoverflow.com\r\n'.encode())
125     connectionSocket.send('\r\n'.encode())
126     print("stackoverflow.com website successfully connected\r\n")
127     break
128
129 # opening pifa website
130 elif response == "/rt":
131     while True:
132         for c in check:
133             elif response == "/so":
```

Run main

Sec-Fetch-Mode: navigate  
Sec-Fetch-User: ?1  
Sec-Fetch-Dest: document  
Accept-Encoding: gzip, deflate, br  
Accept-Language: en-US,en;q=0.9,ar;q=0.8,ar-IL;q=0.7  
Cookie: Pycharm-19fc470a-388e3681-9d5f-4b1c-9f15-b52b533fd0e5

stackoverflow.com website successfully connected

Got connection from IP: 127.0.0.1, Port: 62589





c) If the request is /rt then redirect to ritaj website:

```

123 connectionSocket.send('Content-Type: text/html; charset=utf-8\r\n'.encode())
124 connectionSocket.send('Location:http://www.stackoverflow.com\r\n'.encode())
125 connectionSocket.send('\r\n'.encode())
126 print("stackoverflow.com website successfully connected\r\n")
127 break
128
129 # opening ritaj website
130 elif response == "/rt":
131     connectionSocket.send('HTTP/1.1 307 Temporary Redirect \r\n'.encode())
132     connectionSocket.send('Content-Type: text/html; charset=utf-8\r\n'.encode())
133     connectionSocket.send('Location:https://ritaj.birzeit.edu/register/\r\n'.encode())
134     connectionSocket.send('\r\n'.encode())
135     print("ritaj.com website successfully connected\r\n")
136     break
137
138 # for sending error page
139 while True:
140     for c in check:
141         elif response == "/so"

```

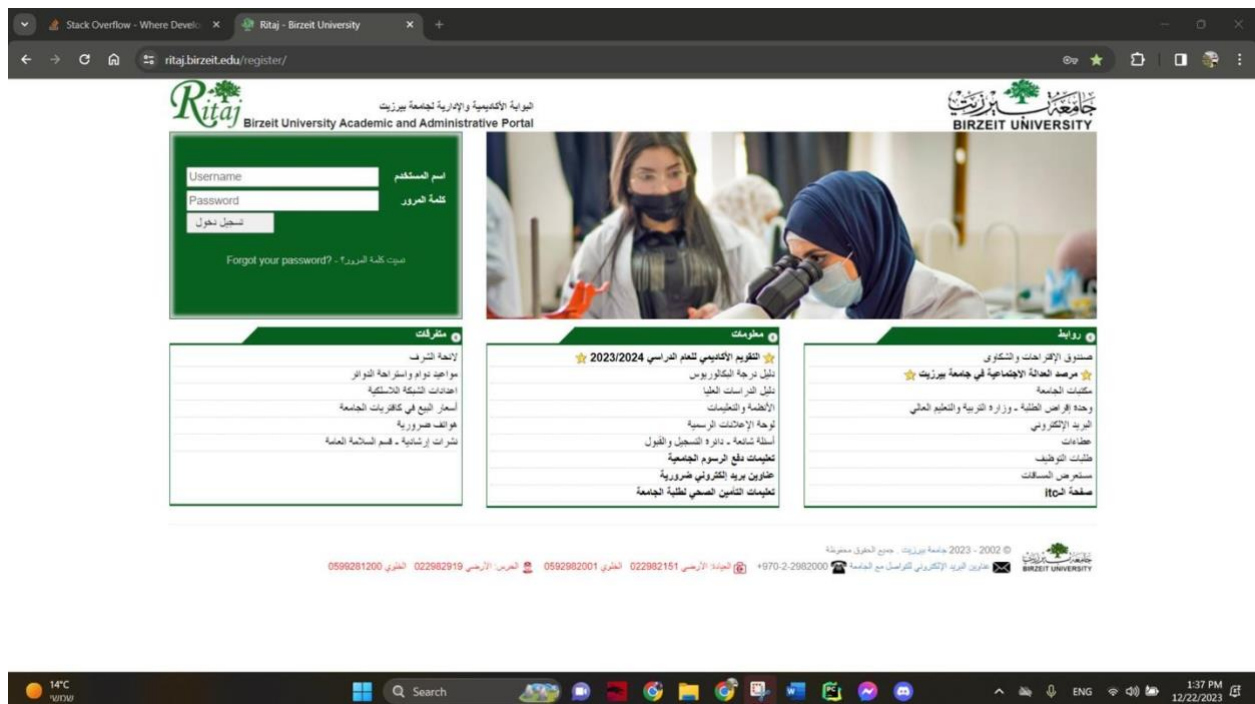
Run console output:

```

Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,ar;q=0.8,ar-IL;q=0.7
Cookie: Pycharm=19fc470a+388e3681-9d5f-4b1c-9f15-b52b533fd0e5

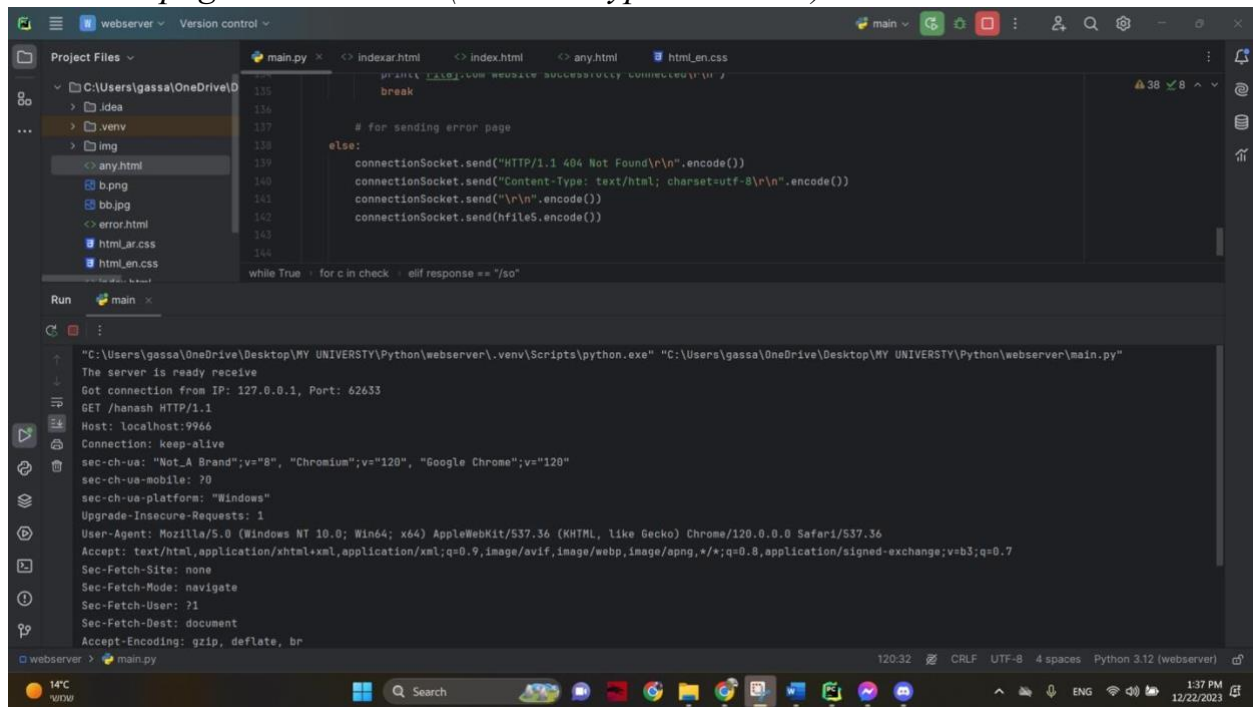
ritaj.com website successfully connected
Got connection from IP: 127.0.0.1, Port: 62607

```





8) If the request is wrong or the file doesn't exist the server should return a simple HTML webpage that contains (Content-Type: text/html):



```
main.py |> index.html |> index.html |> any.html |> html_en.css
135 |> print(f"HTTP/1.1 200 OK")
136 |> break
137 |>
138 |> # for sending error page
139 |> else:
140 |>     connectionSocket.send("HTTP/1.1 404 Not Found\r\n".encode())
141 |>     connectionSocket.send("Content-Type: text/html; charset=utf-8\r\n".encode())
142 |>     connectionSocket.send("\r\n".encode())
143 |>     connectionSocket.send(hfile5.encode())
144 |>
while True:
    for c in check:
        if response == "/so"
```

Run |> main.py

"C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\webserver\venv\Scripts\python.exe" "C:\Users\gassa\OneDrive\Desktop\MY UNIVERSITY\Python\webserver\main.py"

The server is ready receive

Got connection from IP: 127.0.0.1, Port: 62633

GET /hanash HTTP/1.1

Host: localhost:9966

Connection: keep-alive

sec-ch-ua: "Not\_A\_Brand";v="8", "Chromium";v="120", "Google Chrome";v="120"

sec-ch-ua-mobile: ?0

sec-ch-ua-platform: "Windows"

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7

Sec-Fetch-Site: none

Sec-Fetch-Mode: navigate

Sec-Fetch-User: ?1

Sec-Fetch-Dest: document

Accept-Encoding: gzip, deflate, br



Alhasan Manasra 1211705  
Ghassan Qandeel 1212397  
Aws Shaheen 1212585

*HTTP/1.1 404 Not Found*

**The file is not found**



## *Python code*

```
from socket import *
import re
#server #define
server port
serverPort = 9966
#define TCP server
serverSocket = socket(AF_INET, SOCK_STREAM) #
make binding with any ip address by ''
serverSocket.bind(('', serverPort))
# listen for requests
serverSocket.listen(1) # print for
the server is ready print('The
server is ready receive')

# open index.html file with open('index.html', 'r',
encoding='utf-8') as file:
    hfile = file.read()

# open arabic index.html file with open('indexar.html',
'r', encoding='utf-8') as file0:
    hfile0 = file0.read()

# open html file for display it with open('any.html',
'r', encoding='utf-8') as file1:
    hfile1 = file1.read()

# open css file for display it with open('html_en.css',
'r', encoding='utf-8') as file2:
    hfile2 = file2.read()

# open image with png tybe with
open('b.png', 'rb') as file3:
    hfile3=file3.read()

# open image with jpg tybe with
open('bb.jpg', 'rb') as file4:
    hfile4 = file4.read()

# open html for error page with open('error.html', 'r',
encoding='utf-8') as file5:
    hfile5 = file5.read()
    with open('img/gf.jpg', 'rb') as
file6:
        hfile6 = file6.read()
    with open('img/hf.jpg', 'rb') as
file7:
        hfile7 = file7.read()
    with open('img/af.jpg', 'rb') as
file8:
        hfile8 = file8.read()
    with open('html_ar.css', 'r', encoding='utf-8') as
file9:
        hfile9 = file9.read()
```

```

while
True:

    # accept all requests
    connectionSocket, addr = serverSocket.accept()
    ip = addr[0]
    port = addr[1]
    print('Got connection from', "IP: " + ip + ", Port: " + str(port))

    # receive http request and store it in sentence
    sentence = connectionSocket.recv(4096).decode()

    # we split the request to get the request line from user input
    match=re.split(pattern=" ",string=sentence)      response=match[1]
    print("....."+response+".....")      # print http request in
    termanel      print(sentence)

    # for any of this we handle the request depending on the url
    # for sending our html file
    if(response=="/" or response=="/index.html" or response=="/main_en.html" or
response=="/en"):
        connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
        connectionSocket.send("Content-Type: text/html; charset=utf-
8\r\n".encode())
        connectionSocket.send("\r\n".encode())
        connectionSocket.send(hfile.encode())

    elif(response=="/img/gf.jpg"):
        connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
        connectionSocket.send("Content-Type: image/jpg;\r\n".encode())
        connectionSocket.send("\r\n".encode())      connectionSocket.send(hfile6)
        elif (response ==
"/img/hf.jpg"):
            connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
            connectionSocket.send("Content-Type: image/jpg;\r\n".encode())
            connectionSocket.send("\r\n".encode())      connectionSocket.send(hfile7)
            elif (response ==
"/img/af.jpg"):
                connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
                connectionSocket.send("Content-Type: image/jpg;\r\n".encode())
                connectionSocket.send("\r\n".encode())      connectionSocket.send(hfile8)
                elif (response ==
"/bb.jpg"):
                    connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
                    connectionSocket.send("Content-Type: image/jpg;\r\n".encode())
                    connectionSocket.send("\r\n".encode())      connectionSocket.send(hfile4)
                    elif (response ==
"/b.png"):
                        connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
                        connectionSocket.send("Content-Type: image/png;\r\n".encode())
                        connectionSocket.send("\r\n".encode())      connectionSocket.send(hfile3)

    # for sending arabic page
    elif(response=="/ar"):
        connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
        connectionSocket.send("Content-Type: text/html; charset=utf-

```

```

8\r\n".encode())
    connectionSocket.send("\r\n".encode())
connectionSocket.send(hfile0.encode())

    # for sending html file and display it as request
elif(response=="/.html%20file"):
    connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
connectionSocket.send("Content-Type: text/html; charset=utf-
8\r\n".encode())
    connectionSocket.send("\r\n".encode())
connectionSocket.send(hfile1.encode())
    elif (response ==
"/html_en.css"):
    connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
connectionSocket.send("Content-Type: text/css;\r\n".encode())
connectionSocket.send("\r\n".encode())
connectionSocket.send(hfile2.encode())
    elif (response ==
"/html_ar.css"):
    connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
connectionSocket.send("Content-Type: text/css;\r\n".encode())
connectionSocket.send("\r\n".encode())
connectionSocket.send(hfile9.encode())

    # for sending css code
elif(response=="/.css"):
    connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
connectionSocket.send("Content-Type: text/css;\r\n".encode())
connectionSocket.send("\r\n".encode())
connectionSocket.send(hfile2.encode())

    # for sending image with png type
elif(response=="/.png"):
    connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
connectionSocket.send("Content-Type: image/png;\r\n".encode())
connectionSocket.send("\r\n".encode())
connectionSocket.send(hfile3)

    # for sending image with jpg type
elif (response=="/.jpg"):
    connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
connectionSocket.send("Content-Type: image/jpg;\r\n".encode())
connectionSocket.send("\r\n".encode())
connectionSocket.send(hfile4)

    # opening cornell website
elif response == "/cr":
    connectionSocket.send("HTTP/1.1 307 Temporary Redirect
\r\n".encode())
    connectionSocket.send('Content-Type: text/html; charset=utf-
8\r\n'.encode())
connectionSocket.send("Location:

```

```

https://www.cornell.edu\r\n".encode())
connectionSocket.send('\r\n'.encode())
    print("cornell.edu website successfully connected\r\n")
break

    # opening stackoverflow website
elif response == "/so":
    connectionSocket.send("HTTP/1.1 307 Temporary Redirect
\r\n".encode())
    connectionSocket.send('Content-Type: text/html; charset=utf-
8\r\n'.encode())
    connectionSocket.send("Location:http://www.stackoverflow.com\r\n".encode())
    connectionSocket.send('\r\n'.encode())
    print("stackoverflow.com website successfully connected\r\n")
break

    # opening ritaj website
elif response == "/rt":
    connectionSocket.send("HTTP/1.1 307 Temporary Redirect
\r\n".encode())
    connectionSocket.send('Content-Type: text/html; charset=utf-
8\r\n'.encode())
    connectionSocket.send("Location:https://ritaj.birzeit.edu/register/\r\n".enco
de())
    connectionSocket.send('\r\n'.encode())
    print("ritaj.com website successfully connected\r\n")
break

else:
    print("hhh "+response+"hhh")
    connectionSocket.send("HTTP/1.1 404 Not Found\r\n".encode())
    connectionSocket.send("Content-Type: text/html; charset=utf-
8\r\n".encode())
    connectionSocket.send("\r\n".encode())
    connectionSocket.send(hfile5.encode())

```

## Html code

```
<!DOCTYPE
html>
<html lang="en">
  <head>

    <meta charset="UTF-8">

    <!--for link logo tab -->
    <link rel="icon" href="b.png">

    <!--put our tittle -->
    <title>ENCS3320-My Tiny Webserver 23/24</title>
    <!--for link our css code -->
    <link rel="stylesheet" href="html_en.css">
  </head>
  <body>

    <!-- put welcome message with background and logo with diffrent colors ->
    <h1 class="title">Welcome to our course <strong style="color: blue;padding-left:
4px"> Computer Networks, This is a tiny webserver </strong></h1>

    <!-- defining boxes to put our information and detalis -->
    <div class="box">
      
      <div class="student-info">
        <h2> Ghassan 1212397</h2>
        <p> Hay, I am study computer engineering in Birzeit University, i'm
done 76 academic hours in third year , to be honest in last three weeks im
dont see my friend as usually from projects,homeworks and exams ,it is make me
busy and cant play my video games, i wish to be it worth.    </p>
      </div>
    </div>
    <div class="box">
      
      <div class="student-info">
        <h2>hassan 1211705</h2>
        <p> I am currently studying computer engineering in my third year.
Throughout my academic journey, I have successfully completed many projects,
and assignments in my academic subjects. When I have some free time, I love
indulging in my favorite pastimes of watching and playing football games and
going to the gym.</p>
      </div>
    </div>
    <div class="box" >
      
      <div class="student-info">
        <h2>Aws Shaheen 1212585</h2>
        <p>This is my third year majoring in computer engineering at birzeit
university.I am intrested in web development and I am working on bettering my
skills in it, this is one of many projects I worked on for university for
many subjects. In my free time I usually go to the gym and spend time with my
freinds.</p>    </div>
    </div>
```

```

    <!-- defining box to put summary of point 0-->
<div class="box">
    <div class="student-info">
        <h2>point 0 summary</h2>
        <p> in point 0 we open the link and conclude the content type is
doc/html and we need it to make the client or server what tybe of content we
send </p>
    </div>
</div>

<!--links for w3schools and local html file -->
<a href="https://www.w3schools.com/python/python_strings.asp"
target="_blank">W3School</a>
    <p></p>
    <a href="C:\Users\gassa\OneDrive\Desktop\MY
UNIVERSITY\Python\webserver\index.html" target="_blank">local html file</a>
    </body>
</html>

```

## Css code

```
/* define to be our texts nice */
.student-info {
display: flex;                justify-
content: flex-start;         align-
items: flex-start;           /* gap:
16px; */                      flex-direction:
column;

}
/*defining for tittle with background */
.title {
background: burlywood;
width: 1500px;                height:
100px;                        display: flex;
justify-content: center;
align-items: center;
margin-bottom: 32px;
font-family: verdana;
font-size: 20px;
}
/* define to be our texts nice */
h2{
font-style: oblique;
font-weight: bolder;
margin-top: 5px; /* Adjust the margin as needed */
padding: 0;
margin-bottom: 0; /* Adjust the margin as needed */

}

/* Define styles for the boxes */
.box {
display: flex;

border: 1px solid #fffcfc;
padding: 20px;                margin:
10px;                         box-sizing: border-
box;                          position: relative;
overflow: hidden;             gap:
16px;                         border-radius: 16px;
z-index: 1;
}
/* defining to be the text clear */
.box::before {
content: "";
position: absolute;
top: 0;                       left: 0;
width: 100%;
height: 100%;
background-image: url('bb.jpg');
background-repeat: no-repeat;
background-size: cover;
filter: blur(200px);          z-index: -
1;
}
```



```

    }

    /*defining for images sizes */
    .imgg {
width: 150px;
border-radius: 16px;

    }
    /*defining for be elemants horizontal */
    .horizontal-container {
display: flex;           flex-direction:
row;
    }

    /*defining for background of body */
body {
    background-image:
url("bb.jpg");           background-
repeat: no-repeat;       background-
size: cover;
    }

    /* define to be our texts nice */
h1{
    text-align: center;
    font-family: 'Times New Roman', Times, serif;
    }

    /* define to be our texts nice */
p {
    font-family: 'Times New Roman', Times, serif;    font-
size: 25px;
    margin-top: 0; /* Adjust the margin as needed */
margin-bottom: 5px; /* Adjust the margin as needed */
color:#fffcfc;    font-style:inherit;    font-
weight:90;
    }

```